UNIVERSITY OF CALGARY

Continuous Models of Genetic Regulatory Networks and the Multistability Problem

by

Kelly John Rose

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

January, 2009

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Continuous Models of Genetic Regulatory Networks and the Multistability Problem" submitted by Kelly John Rose in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Supervisor, Dr. Larry Bates
Department of Mathematics and Statistics

Co-Supervisor, Dr. Stuart A. Kauffman
Director, Institute for
Biocomplexity and Informatics,
University of Calgary

Dr. Len P. Bos
Department of Mathematics and Statistics

Dr. Robin Cockett
Department of Computer Science

Date

Abstract

Genetic regulatory networks are highly complex dynamical systems which are believed to control the behaviour and evolution of cells within many lifeforms through their generation of mRNA and respective proteins. One of the major problems in mathematical biology, the MULTISTABILITY problem, is the question of whether or not a given genetic regulatory network has more than one stable fixed state. In a way, it provides a method to find switches within the large genetic regulatory network and thus understand how to nudge the state of a cell based on the state of these switches. This thesis studies various models of genetic regulatory networks and develops a new method to approach the MULTISTABILITY problem, called an activation-repression numbering. By using this numbering, the problem is determined to be NP-Complete and thus intractable. However, further study provides a heuristic which can solve the MULTISTABILITY problem for special cases efficiently.

Acknowledgements

I would like to extend the deepest appreciation to my advisors: Stuart Kauffman, for inviting me to Calgary to research theoretical biology with him and giving me an opportunity few mathematics students ever get – a chance to work and research in a advanced biology lab. His enthusiasm and creativity provided a never-ending supply of new and fascinating directions for mathematics as applied to biological and ecological studies. Larry Bates also deserves my gratitude for watching over me and guiding me throughout the thesis process, allowing me considerable freedom, but still providing a gentle and wise guiding hand.

I would like to thank Hilary Carteret, David Foster, Mircea Andrecut, Dennis Salahub, Gordon Chua, Sui Huang, Sergei Noskov, Julie Thoms, Bob Este and the rest of the team at the Institute for Biocomplexity and Informatics at the University of Calgary for their stimulating discussions, effective criticism, and quality companionship during my two years there. With their valuable feedback, knowledge and advice, I was able to understand the rich and complex biological concepts required for this thesis.

I would like to acknowledge my parents, Patrick and Louise Rose, and the rest of the Rose family. Without them and their neverending belief in my abilities, I would never have even made it this far.

Finally and most importantly, I want to thank my wife, Tracy Rose, for being there for me throughout this entire process. She has been there throughout my thesis work, supporting me even when things seemed the hardest, providing the love and care I needed to find the strength to fully develop my work.

Dedication

This thesis is dedicated to my wife, Tracy Cogsdill Rose.
Without her faith in me, none of this would have been possible.

Table of Contents

List of Figures

Epigraph

Essentially, all models are wrong, but some are useful.

George E. P. Box, Norman R. Draper, Empirical Model-Building and Response Surfaces

Chapter 1

Introduction

Mathematical biology is a young but sophisticated branch of mathematics. It takes results from applied and pure mathematics, including dynamical systems [62], combinatorics [75], computational theory [76], topology [54] and statistics [18] and uses them to gain an insight into the complex sciences of biology and medicine. Both are fields which do not easily yield to rigourous mathematical analysis, due to the sheer immensity and complexity of the systems with which they work and the chaotic behaviour thereof. Statistical averaging or simplification of these systems commonly provides faulty deductions, while extensive computational simulation generally requires excessive resources and does not provide much intuition beyond the specific model being simulated. Historically, due to these complexities, biology has been more dependent on empirical techniques than mathematical ones [58]. Traditionally, the only area in which mathematical biology ws commonly used was in population dynamics where statistical averaging techniques could be effectively used to determine the ebb and flow of various population groups [72].

However, with the recent decoding of the human genome [48] and the slow reverse engineering of the genetic regulatory network [69], as well as recent work in neuronal studies, the need for a solid mathematical model to gain intuition on these complex systems has become essential. Billions of dollars of research has been poured into this field in the hopes that the intuition gained from new models and theorems will lead to more and better drugs and treatments. While a majority of the models have been complicated computational models based on simulating the system dynamics, there has been some exciting research in determining methods to simplify said models without sacrificing their deductive capabilities. Within genetic regulatory networks, this has led to models such as Kauffman's Boolean network model [41] and Glass's piecewise linear ODEs [32].

In the human genome, there are over 20,000 protein producing genes [39], with their proteins each interacting with other genes and proteins in surprisingly complicated ways. To define the entire molecular system to a reasonable level of accuracy based on experimentally known results would be an immense project. To perform the experimental work required to determine the entire molecular system beyond known results would require large numbers of labs collaborating for decades. This is all assuming non-genetic interactions are ignored and without discussing the implications of any of the chemical reactions. Obviously, this is effectively useless for providing intuition about potential drug candidates for any disease or disorder, so being able to find the most important reactions and mathematically study their behaviour analytically becomes vital.

One of the first methods to simplify the system was to use a mapping from the continuous real world to a discrete system. Known as Boolean networks, this was done by Kauffman in 1969 [41]. He visualized a gene as either being on or off, instead of as a continuous range. This allows for the interactions between genes to be represented by Boolean equations and for easy simulation of a large number of randomly constructed networks. Using a method he refers to as the ensemble approach, he proceeded to generate large numbers of these networks randomly and with varying network-wide parameters, like the average in-degree, or the Boolean function bias. By examining how the dynamics of these networks changed as these parameters changed, he deduced three realms of discrete dynamics, the ordered, chaotic and critical realms, and deduced correctly that the dynamically critical point occurs with in-degree $k = 2$, a result proven mathematically later by Derrida in the 80s [20].

While this model has the benefits of ease of simulation and simple statistical calculations of the dynamics and damage propagation, it has the major problem that the dynamics are synchronous, noiseless and discrete, three properties that a real world genetic regulatory network simply does not have. While the results regarding them are definitely interesting from the perspective of complexity science, they do not directly map to the real world systems observed. To resolve this, a variety of methods have been developed, both expanding Boolean networks to continuous, noisy or asynchronous systems [32, 63, 35] and simply converting the complex system of chemical equations within the genetic regulatory networks into related systems of dynamical or stochastic equations [59, 4]. The former have the problem that the system remains fairly artificial, and the latter have the problem of commonly being too complex for analysis.

So, if an applied mathematician were to apply results from dynamical systems theory on the entire system by creating a system of ODEs, PDEs or SDEs, they would quickly become overwhelmed by the mass of equations. If they were to use discrete models, only examine small subsections or study the system statistically, they end up missing important dynamics which could be useful. Instead of trying to create a model of the system before working with it, another method is to clearly define a particular problem which needs to be solved and then develop a model which reflects the important dynamics, thus simplifying the system and not missing anything important or major to that particular problem. This thesis attempts to do just this by defining the MULTISTABILITY problem, a useful problem for drug research involving gene-gene interations, mapping the genetic regulatory network into a statistical model of the chemical dynamics and then further mapping that system of ODEs into a clear combinatorial problem to easier analyze said problem.

By defining the problem first, it becomes easier to make sure the model which is being used is capable enough to at least understand the complexity of the problem and the difficulty of solving it. In computer science, Turing Machines were invented with the hope of solving a problem known as the Entscheidungsproblem [15, 73]. After developing a model of computation which was powerful enough to represent real world computation but simple enough to mathematically analyze, this problem ended up being unsolvable.

In the case of the MULTISTABILITY problem, which is the prime focus of this thesis, the model on which the problem is mapped ends up demonstrating that the problem is difficult even with heavy restrictions on the possible networks.

Even though the problem ends up being difficult, not all is lost. By mapping it to a simpler system, it is possible to add intuitive restrictions which allow it to be solved in certain cases and special properties of stability to be teased out of the structure of the networks. The final chapter of this thesis offers some useful theorems showing how with further restrictions it becomes possible to rigourously handle the MULTISTABILITY problem and other related problems for certain genetic regulatory networks. These results could prove useful to find possible drug targets within real-world networks, even if the real-world networks have extra properties which are not considered in the model.

To place the MULTISTABILITY problem into context, it is useful to examine two small real world examples where finding a MULTISTABLE subsystem provided deep insights into human cellular behaviour. The first real world example of a MULTISTABLE subsystem within the human genome which has massive implications for cancer treatment and stem cell therapy is the PU.1, GATA-1 toggle switch within the haemopoetic cell line (blood cells). These two genes, PU.1 and GATA-1, mutually repress each other's expression while simultaneously activating their own. The second real world example where finding a variety of MULTISTABLE subsystems would be useful is the discovery of the switches which can revert a mouse skin cell back to a progenitor cell (ie. a cell that can potentially turn into any other type of cell).

The behaviour of the PU.1, GATA-1 toggle switch is believed to be essential in determining what a haemopoetic stem cell eventually turns into. When a haemopoetic stem cell is still in a progenitor state, the PU.1, GATA-1 toggle switch lies within a very small but stable state where both PU.1 and GATA-1 have low but relatively equal expression. As the cells develop, other genes within the regulatory network adjust the strength of the activation and repression for the toggle switch until their initial state becomes unstable, and the switch must either fall to a PU.1 high, GATA-1 low state, or a PU.1 low, GATA-1 high state. What is fascinating about this switch is that the final state it falls into determines what the haemopoetic cell eventually becomes. If GATA-1 is high, the cell will eventually evolve down the myeloetrythroid cell line becoming erythrocytes or thrombocytes (red blood cells), and if PU.1 is high, the cell evolves down the myelolymphoid cell line becoming granulocytes or macrophages (white blood cells)[40]. By having the knowledge about the switch's existence and how it works, it becomes possible to direct a group of stem cells to become the specific cell needed for treatment [5, 57]. Similarly, it provides a method to force cancerous leukemic stem cells to differentiate into benign blood cells [23].

The PU.1, GATA-1 toggle switch is not the only empirical example of a MULTISTABLE subsystem. In recent experimental work, a team at Kyoto University and a team in Massachusetts succeeded in reprogramming mouse fibroblast (skin) cells to become pluripotent cells. These cells are similar to embryonic stem cells in that they can theoretically differentiate to any other cell type. [55, 77]. This was accomplished by introducing retroviruses designed to overexpress a variety of transcription factors (Oct4, NSox2, c-Myc, etc.) It is believed by some in the field that this overexpression has the effect of resetting a variety of genetic switches (ie. MULTISTABLE subsystems) to revert the cell to a state similar to the embryonic stem cell state. In effect, they reprogram the cells back to their original state. The discovery of this specific combination of genes came from previous work where they overexpressed a variety of genes seen in embryonic and cancerous cells to see if it would generate pluripotent cells [71]. Over the course of 2007, multiple papers came out extending this work to human cells, and improving the technique [78, 70, 53, 9, 51]. At the end of 2007, these pluripotent cells, now referred to as induced pluripotent stem (iPS) cells, were starting to be used to develop experimental treatments for diseases like sickle cell anemia [34].

In Takahashi's original work, far more work was done than was needed to generate pluripotent cells, as Wernig, et al. showed. However, if the MULTISTABILITY problem were to be solved and a requisite genetic regulatory network was known, it may be possible to prune down those and save money and experimental effort. In a similar fashion, in the spirit of the work with the PU.1, GATA-1 toggle switch, finding MULTISTABLE switches within a larger system allows for not only the ability to reprogram a cell, but to push it forward in a manner that a doctor would prefer for a treatment. In the case of a cancer stem cell, push it from its reproductive state into a benign differentiated state, or in the case of pushing a set of pluripotent cells towards a specific tissue type. These examples will be referred to throughout the thesis to relate the mathematics to direct biological consequences.

Chapter 2

Computational Complexity

2.1 Introduction

Informally, an algorithm is a series of instructions which, given a specific input, will process that input in a clearly defined manner and produce a well-defined output. Obviously, this definition lacks the mathematical formalism needed to do any thorough mathematical analysis. The first attempt to formalize the concept of an algorithm was through David Hilbert's Entscheidungsproblem (literally "decision problem"), defined in 1928 [36]. His hope was to either discover an algorithm or prove the non-existence of such an algorithm which, given any description of a formal language and a statement in said language, could determine if such statement is "true" or "false" within the formal language. Any such algorithm could essentially act as a theorem prover. If it was found, it would completely revolutionize mathematics because mathematicians would no longer have to prove theorems, only come up with them and test them using the algorithm. This problem, which Hilbert was unable to answer conclusively, was solved almost a decade later by Church and Turing [15, 73]. They showed that such a generalized algorithm simply could not exist. Their proofs demonstrated that it was impossible to decide algorithmically (ie. using a mechanical computational system) whether any given statement within arithmetic is true or false and thus no such algorithm could exist to deal with the generalized Entsheidungsproblem if the formal system was powerful enough to handle basic arithmetic. Gödel's incompleteness theorem, proven after this point, implies a similar result [33].

From this work, there developed a formal definition of an algorithm. A computational algorithm is a set of instructions which, given a specific initial state, will proceed through a series of well-defined successor states and eventually complete in a final state. Shortly after this mathematical concept of a computational algorithm was developed, similarities between a variety of problems were noted. Some problems ended up being uncomputable, others required very few resources to answer, others required massive amounts, many seemed to be easily transformed into other similar problems. After a while, it became obvious that there seem to be many different classes of problems based on the algorithms that could solve them and the amount of resources required they required, where resources are either processing time or memory space required. In some circumstances, a set of problems could be efficiently solvable, requiring few resources in time or space. In other, less desirable circumstances, a problem would require exponential amounts of both resources. In the worst circumstances, as it is with the Entsheidungsproblem, there simply may not even exist an algorithm to solve it at all

2.1.1 Big-O and Big-Ω notation

Computational complexity theory developed out of this attempt to determine the amount of resources needed to solve a class of problems. The time (space) complexity of a problem is commonly defined as the time (space) requirements for a machine to solve said problem using a given computational model as a function of the input size for the worst-case instance of the problem being solved. The input size of the instance is commonly defined as the number of characters in the formal language it requires to define an instance of the problem completely. As this is rarely computable exactly, a type of notation known as Big-O notation is used. This allows for the average runtime of the function to be understood not as how many resources it takes for any given instance of the problem, but rather how the need for resources grows as the instance size approaches infinity.

Historically, big-O notation existed before the creation of computational complexity theory. It proved very useful in analytic number theory to simplify equations, especially those with regards to errors in calculation. The first known presentation was by Bachmann in the second volume of his book on analytic number theory [6]. About a decade and a half later, Landau popularized it in his textbook on the math regarding prime numbers and hence is sometimes referred to as either Bachmann-Landau notation or simply Landau notation [47]. With regards to computational theory, Donald Knuth provides a useful discussion of this notation and other such asymptotic notation in the "asymptotic representations" section of the first volume of his work, "The Art of Computer Programming" [45].

Big-O Notation We define the set $O(f(x)), x \in \mathbb{Z}$, as follows

$$g(x) \in O(f(x)) \text{ as } x \to \infty \Leftrightarrow \exists n_0, M \geq 0 \text{ such that } |g(n)| \leq M|f(n)|, \forall n \geq n_0$$

It is important to note that big-O notation can also be defined for $x \to a$ for any $a \in \mathbb{R}$. However, for the purposes of this thesis, this extra detail is not necessary. Thus, when this thesis states $g(x) \in O(f(x))$, it is implied that this is for $x \to \infty$. Another important note must be made about the abuse of notation common within computational complexity literature. While this thesis will use the correct notation, in many of the papers in the literature, it is said incorrectly that $g(x) = O(f(x))$ when it should be $g(x) \in O(f(x))$. Also, the use of big-O notation is helpful for algorithmic analysis because the given big-O class of a function used for a time (space) complexity of an problem can simply be based on the most efficient algorithm known for said problem. This makes sense since big-O notation provides an upper-bound on the complexity of a given problem. Another notation, big-Ω notation, is used for asymptotically defining a lower-bound complexity for a problem. Unfortunately, these lower-bounds seems to be much harder to prove rigourously for many problems.

Big-Ω Notation We define the set $\Omega(f(x)), x \in \mathbb{Z}$, as follows

$$g(x) \in \Omega(f(x)) \text{ as } x \to \infty \Leftrightarrow \exists n_0, M > 0 \text{ such that } |g(n)| \geq |Mf(n)| \forall n \geq n_0$$

Again, the same notational issues exist for this class of functions within the literature as for the big-O class from above.

A commonly used example of a problem with a lower bound for complexity is the sorting problem, given an unordered set S of elements with an ordering relationship $(\leq)$, find a ordering $T = t_0, t_1, ..., t_n$, such that $t_i \leq t_{i+1}$, $\forall t_i \in S$. In this circumstance, when the computational model is restricted to comparison based sorts, any algorithm which can solve the problem will have runtime in $\Omega(n \log n)$. However, if the set being sorted has extra properties such as the ability to sort lexicographically, it is possible to decrease this runtime. For example, Radix sort can sort words or integers in $O(n)$ if they have a fixed maximum length. [46]

With the Big-O notation, and to a lesser degree, big-Ω notation, it becomes possible to clearly define the relative difficulty of any given problem given a specific computational model without having an exact solution. This is because big-O provides an easy-to-find upper bound (the most efficient known algorithm), and the big-Ω provides a lower bound, if one can be found.

2.2 Decision Problems

A major class of problems studied in computational complexity theory are problems for which, given an instance, any algorithm will return either “accept” or “reject”. In simpler terms, these are problems for which, given an instance of the problem, the answer is either yes or no. In computational complexity, these problems are known as decision problems. As an example, the problem 'given two graphs G, H, are the graphs isomorphic?', is a decision problem. In contrast, the problem, 'given two graphs G, H, find an isomorphism, if one exists,' would not be a decision problem, rather it would be what is known as a function problem, since the answer would be the isomorphism itself. It is important to note that while a solution to the function problem can possibly answer a related decision problem, the same is almost never true in reverse. It is entirely plausible that an algorithm could exist to solve the decision problem which provides no insight whatsoever towards solving an associated function problem.

Traditionally, a decision problem P is said to be equivalent to the set of inputs I for P in which the given answer is a yes. This formally means that a decision problem can be represented as a subset of the natural numbers, where the problem is rewritten as “is a given number an element of this set?” For a decision problem where the instances are not immediately natural numbers the equivalency is simple: Take an instance I of a decision problem, and define an encoding onto the natural numbers, most commonly by using a Gödel numbering. This turns the set of instances for which the given answer is yes into a subset of the natural numbers. [33]

A couple of examples of well known decision problems include the satisfiability problem and the halting problem. The satisfiability problem is a difficult problem which will be discussed later in this chapter. The halting problem is the problem of determining if any given program will eventually halt or not on a given computational model. It is similar to the Entscheidungsproblem in that it is also an uncomputable problem. This is easily seen by the fact that if the halting problem had an algorithm that could solve it in the general case, the question of what it would do when given itself as input is paradoxical. Another decision problem on which this thesis will be mostly concentrating is the MULTISTABILITY problem from genetic regulatory network studies.

2.2.1 The MULTISTABILITY Problem

The MULTISTABILITY problem is an example of a decision problem which arises in mathematical biology. Informally, the problem is, “given a specific genetic regulatory network, does there exist two stable states for which the network remains fixed.” As will be discussed in the MULTISTABILITY chapter, this is not a well-defined problem since the model used for the genetic regulatory network is not defined, and neither is the definition of a “stable fixed state.” Thus, as is done with almost every other real-world problem, the system is mapped onto a reasonable model for which the problem can be well-defined, an algorithm is built on the model and the results produced by the algorithm are mapped back onto the real-world system and empirically verified. This has been done successfully with many different models of genetic regulatory networks, especially the regulatory Boolean network model and the IADGRN [43] and ARACNE [52] algorithms for reverse engineering them.

Any model on which a problem is solved will naturally have some discrepancies with real-world phenomena. For example, the regulatory Boolean network model can make errors with regards to stable fixed states in very special cases when it is compared to some of the continuous models. This is because a fixed point in a Boolean network is inherently a stable fixed point since it is impossible to have an ϵ perturbation of the discrete state values; only a full inversion of any node’s Boolean value is possible. However, in a continuous model the fixed points can be both stable and unstable. Since these models generally represent real-world interactions more closely, this leads to instances where the Boolean network MULTISTABILITY problem would give a yes answer, while the real-world network upon which the Boolean network is constructed does not have these stable fixed points. Naturally, if this problem became endemic, it would make it difficult to apply any algorithm based on the this model to real-world biological science or medicine.

Why is solving this problem useful? If the model is good enough, it provides a way to scan through a given genetic regulatory network for subnetworks that potentially act as important switches. A good idea of which subnetworks within a larger network could possibly act as switches would provide the ability to target said networks with drugs or another appropriate differentiation therapy. It is believed that these multistable “switches” are important for determining cell types, maintaining cell state, and could perhaps be even involved with the creation and evolution of cancer. This was seen with the PU.1 and GATA-1 switch for haemopoetic cell linages. The ability to find them rapidly and target them would be priceless for finding potential drug candidates, as well as provide a new method to determine the current state of a given cell without using morphometric parameters. Similarly, it would also be useful for stem cell research by helping direct differentiation down desired pathways to produce the desired tissues or organs for transplant.

It is important to note that any solution to the MULTISTABILITY problem would only tell whether or not multiple stable states exist. A solution to the problem of defining what the stable fixed points are for a given genetic regulatory network would naturally be even more valuable. However, even though this is more useful than the simple decision problem, it adds a significant level of complexity to the problem and thus any algorithm built to solve it. There do exist some algorithms that perform similar tasks but they are either not tractable for large networks (the best known algorithms run in $O(1.19^n)$ time, or only work efficiently on a small subset of the problem space [21, 79]. Since the decision problem is simpler and no polynomial time algorithm is known for it, the complexity of the decision problem and some heuristics to approach it are examined in this thesis.

2.3 Turing Machines

To clearly define the important complexity classes for this thesis, it is necessary to define the abstract computational model upon which the algorithms will be run, and with that model, define the concepts of space usage and time required to run the algorithm. There are a variety of abstract computational models which can be used, but the most popular of them is currently the Turing machine model. In this thesis, this will be the primary model used when a model is needed. Created in 1936 by Alan Turing [73], it was intended not to be constructed as described but rather a theoretical model designed to find the limits of what can be effectively computed mechanically. He hoped to create what is referred to as a "universal computer," a computer which is capable not only of running algorithms, but simulating any other machine that can also run algorithms. He described this idea in his paper [73].

> It is possible to invent a single machine which can be used to compute any computable sequence. If this machine U is supplied with the tape on the beginning of which is written the string of quintuples separated by semicolons of some computing machine M, then U will compute the same sequence as M.

In this statement is the essence of the Church-Turing thesis. Turing believed, and it is now generally accepted, that any computational machine can be simulated by this Turing Machine concept. Thus, his machine is literally a universal computing machine. In its time, this was a radical concept. However, today this fact is accepted even though it has not been and may never be rigourously proven. He had developed a machine which was simple enough to be abstracted out with mathematical rigour but believed to be capable of doing any computation that any real-world machine could do.

The original machine Turing described was not originally referred to as a deterministic Turing machine. However, as time progressed, other types of Turing machines were developed which changed some aspects of the original Turing machine in subtle ways in order to achieve higher efficiency or in an attempt to disprove the Church-Turing hypothesis. In almost all cases, the deterministic nature of the machine was modified in some fashion. Thus, Turing's original design is now more accurately referred to as a "deterministic Turing machine," in reference to the deterministic nature of the transition function used to determine the next action for the machine.

As noted, there are many variations of Turing machines, such as the “non-deterministic Turing machine” which has the property that the transition function is a multivalued function. It is known that a “non-deterministic Turing machine” can be simulated by a “deterministic Turing machine.” Another common type of Turing machine is the “probabilistic Turing machine." This machine has the property that it chooses between transition functions based on some probability distribution. If it is accepted that deterministic algorithms exist which can generate perfectly random numbers, then these can also be simulated on a deterministic Turing machine. In both cases, the original Turing Machine remains universal, if somewhat less efficient in resource use than its counterparts [56]. Even a Turing machine based on quantum mechanical principles can be simulated by a deterministic Turing machine given enough resources [12].

2.3.1 Deterministic Turing Machines

A deterministic Turing machine is what most people think of when they refer to a classical Turing machine. Mechanically, a Turing machine is a machine consisting of four primary parts:

1. A table of instructions which tell the machine what to do given the state that it currently is in.
2. A tape of infinite length in both directions.
3. A head which can move back and forth on tapes and read and write symbols.
4. A state register which stores the current "state" of the Turing machine.

The number of possible states is finite, as is the language of symbols that can be written on the tape. However, the tape itself is infinite. Thus, even though the number of states and symbols is finite, it remains possible for a program to run forever without ever having the Turing machine, as a whole, repeat itself. Figure 2.1 shows a simplified representation of the essential mechanical parts of a deterministic Turing machine.

Mathematically, a deterministic Turing machine is defined as a 6-tuple, $M = (Q, \epsilon, \Sigma, \delta, q_0, F)$ where each of the objects are defined as follows. This definition is similar to Hopcroft and

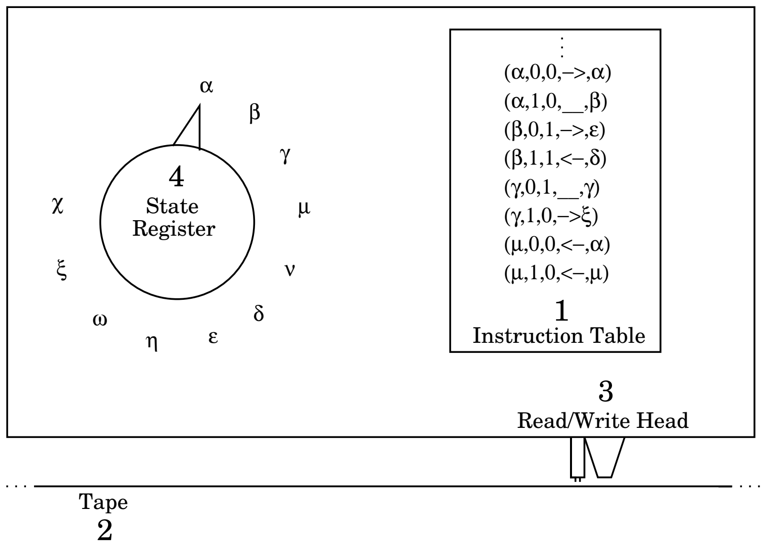


Figure 2.1: A deterministic Turing machine

Ullman's definition. However, this definition notably omits the "set of input symbols" which is the computer alphabet less the blank symbol [37].

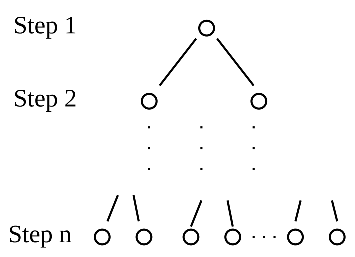
1. Q is a finite set of states
2. Σ is the computer's alphabet
3. $\epsilon \in \Sigma$ is the "blank symbol"
4. $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, N\}$ is the transition function, where $L =$ move tape left, $R =$ move tape right, $N =$ no shift.
5. q_0 is the initial state
6. $F \subseteq Q$ is the set of possible final sets (ie. the states where the computation stops and returns some final value.)

Since a Turing machine is believed to be universal by the Church-Turing thesis, any mechanical computation which can be performed is believed to be possible on a Turing machine. However, any calculation may not be as efficiently computed when it is simulated on a Turing machine. Instead of dealing with efficiency, the Turing Machine provides a tool to determine if it is even possible to perform a certain computation mechanically, since if the Church-Turing thesis is believed, then if it's impossible to do with a Turing machine, it's impossible to do. While it is not the most efficient of mechanical computation devices, at this point in time, no known physical device not using quantum mechanics is capable of operating more efficiently than a deterministic Turing machine in any significant asymptotic fashion on the size of the input. So, even though there exist theoretical devices which seem to be more efficient, the Turing machine is a good model of computation for determining the relative difficulty of computing solutions to problems.

2.3.2 Non-deterministic Turing Machines

A non-deterministic Turing machine is another theoretical construction of a computational device. The primary difference between a deterministic Turing Machine and its non-deterministic counterpart is that the transition function of a non-deterministic Turing machine may be multivalued. Physically, it is not easy to intuit what this means. Some academics describe the machine as a branching process where every time a transition function has more than one output, the computer makes multiple copies of itself each following one of the outputs. Thus, while a deterministic Turing machine has a clear computational path, a non-deterministic Turing machine has what could better called a computational tree. This can be seen in figure 2.2. In the figures, each edge represents a path for the transition function, and each node the current set of states for the machine. While this description is apt, another somewhat more intuitive way of understanding a non-deterministic turing machine is to say the machine is a "perfect guesser" and must always choose the transition which leads to an accepting state, if one exists. If one does not exist, it chooses any random transition [28].

As discussed before, this Turing machine is computationally equivalent to the deterministic Turing machine, since a deterministic Turing machine can simulate a non-



(a) deterministic (b) non-deterministic

Figure 2.2: A deterministic computation path and a non-deterministic computation tree

deterministic Turing machine and vice versa. However, a large class of problems are believed to be more efficiently solved on this machine than on a deterministic Turing machine. While Savitch demonstrated that any simulation of a non-deterministic Turing machine by a deterministic one will require at most a square of the non-deterministic space requirements [61], it is believed that any simulation will require exponential time. However, this belief has not been formally proven. A simple, however non-rigourous, argument in favour of this conjecture is the fact that a nondeterministic Turing machine is able to search an indexed set of size 2^n in n steps, by utilizing the branching nature of its processes.

Non-deterministic Turing Machines have another property in that their non-determinism is bounded non-determinism. This implies that the multivalued function has a finite number of possible transitions. This is inherent in the fact that the transition function can only have a finite set of values since the alphabet, states, and possible head movements are all finite. This is fundamentally important for the non-deterministic turing machine to be realistic since if the non-determinism were unbounded, the halting problem would become solvable [67].

2.4 Complexity Classes

A complexity class is a group of problems which are solvable given a specific Turing machine model and a limited amount of resources based on the input size of any given instance. There is an entire zoo of complexity classes, each of which is interesting in its own right [1]. However, this thesis will concentrate on the most common and popular complexity classes: P (Polynomial), NP (Non-deterministic Polynomial) and PSPACE (Polynomial Space) and the various classes associated with them.

2.4.1 Polynomial Reduction

Polynomial reduction is a mathematical ordering which can exist between two given problems. Sometimes it is possible to take an instance of one problem and transform it into an instance of a more easily solved problem (or a problem for which an algorithm already exists). Now, if such a transformation requires very few resources, then the difficulty of the first problem can be seen as being at most as difficult as that of the second problem into which it is transformed, since a solution to the second problem would imply a solution to the first problem, and not vice versa.

Formally, given two decision problems $S, T, S \leq_P T$ or, equivalently, S polynomially reduces to T, if there exists an algorithm f requiring polynomial resources (i.e., its runtime is in $O(p(n))$ for some polynomial $p(n)$, which takes an instance $i \in S$ and maps it to an instance $j \in T$, such that i is true for S, if and only if $j = f(i)$ is true for T. It is useful to note that this need only be a one-way function and thus a one-way reduction.

2.4.2 P

The acronym P stands for “deterministic Polynomial,” or simply “Polynomial.” It is the class of problems for which there exists an algorithm which can solve the problem using a polynomial number of resources on a deterministic Turing machine. This implies that given an instance I of size $|I| = n$, there exists an algorithm for which the runtime will be $f(n) \in O(p(n))$ for some polynomial $p(n)$. Note that this does not immediately imply the problem is easily solved, as for large instances the polynomials $p(n), q(n)$ can be of very high degree. It does, however, imply that the difficulty of the problem does not grow exponentially on the size of the instance, which means that the computational resources needed are likely to be reasonable. While not entirely accurate, if a problem lies within P, it is said to be an easy or efficiently-solvable problem.

Some examples of problems that lie within P include finding the shortest distance in a graph [22], determining if a number is prime [2], linear programming [44] or finding a maximum matching [24]. For all of these problems, there exist known algorithms for which the resource usage is polynomial. Most problems known to lie in P have this property. Yet, there do exist a few problems for which a polynomial time algorithm has been proven to exist, but no algorithm is currently known, i.e., a non-constructive proof of the existence of such an algorithm is known, but the exact algorithm which is polynomial in resources is not. For example, the Robertson-Seymour theorem [49] implies the existence of an algorithm with runtime $O(n^3)$ for determining if a graph is part of a given downwardly closed set of graphs, such as planar graphs on the surfaces of various objects, will exist [8]. However, the actual algorithm for any given set is not immediately known.

In comparison, an example of a problem known not to lie in P is the problem of whether a given algorithm will halt on a Turing machine in k steps or less; this problem is known as the "bounded halting" problem [50]. This lies in a complexity class known as EXPTIME, the class of problems for which it takes an exponential amount of time given the input to run, and not P. To see this, it is important to remember that the runtime is given as a function on the size of a given instance. Since the number k can be encoded in $log(k)$ bits, the instance size is $|I| \in O(log(k))$. To answer the "bounded halting" problem, the algorithm is run for k steps, and it is determined if it halts. Since it requires $O(2^{|I|})$ time, an exponential time is required. Thus, it lies in EXPTIME and not in P.

2.4.3 NP, NP-Complete, and coNP

The acronym NP stands for "Non-deterministic Polynomial". It is the class of problems for which there exists an algorithm which can solve the problem using a polynomial number of resources on a non-deterministic Turing machine. As with the P complexity class, this implies that given an instance I of size n, there exists an algorithm for which the runtime is $f(n) \in O(p(n))$ for some polynomial $p(n)$. Contrary to popular mathematical parlance and myth, this does not imply that the problem lies in "not-polynomial." If it were ever proven that a problem lying in NP will have all algorithms' runtimes in a big-Ω class like $\Omega(2^n)$, then this would be resolved as it would imply $P \neq NP$. Similarly, if a special type of NP problem called a NP-Complete problem were to lie in P as well, then $P = NP$ [16]. There has been some work on this problem using models where it is assumed a specific polynomial time algorithm exists, without defining said algorithm. For some models, $P = NP$ and for others, $P \neq NP$ [7]. However, at this time, there has been little clear progress in either proving or disproving this conjecture in the general case.

When a problem C is said to be "complete" in its complexity class, it means that there exists a polynomial reduction from any other problem P in that class to C. Thus, if a polynomial algorithm is known to solve C, a polynomial algorithm can be derived to solve any problem P in the entire complexity class. Commonly, given a complexity class K, the set of all complete problems in that class form a class known as K-Complete. So, for NP, the set of all complete problems in NP form the class NP-Complete [28]. It is useful to note that there do exist complexity classes for which no there are no complete problems [65].

The current believed relations between P and NP can be seen in figure 2.3. In it, all problems contained within P are inherently contained within NP as a non-deterministic Turing machine can do everything a deterministic one can, and it is assumed that $NP \neq P$. This is from the fact that there are many well studied problems contained within NP for which no polynomial algorithm is known to exist, such as the graph isomorphism problem, the Boolean satisfiability problem and the Hamiltonian path problem, the problem of finding a tour in a graph which touches all vertices only once [28]. This seems to imply a high level of difficulty for problems which are NP and have not yet succumbed to a polynomial algorithm.

From an efficiency standpoint, it is easy to see that if a problem is complete within its complexity class, then any algorithm which solves said problem must be at least as time or space intensive as any other problem within said complexity class. For this reason, if a problem lies within the class of complete problems in NP, NP-Complete, then it is generally believed to not have a polynomial time algorithm. This is regardless of the lack of a formal proof that $P \neq NP$. NP-Complete contains such a wide variety of difficult decision problems which have not succumbed to any attempt to find a polynomial algorithm, that it is generally accepted if a problem is proven to be NP-Complete then the problem is not tractable in its current form. When this occurs, most experts resort to heuristics to find approximate solutions or restrictions of the original problem space in the hopes that by restricting it the problem will make it tractable.

A special property of the NP complexity class is that for any given decision problem, if the answer is affirmative, there will exist a polynomially sized "proof" which can be checked in polynomial time [28]. From the perspective of the process tree method of understanding non-deterministic Turing machines, the "proof" is the specific set of branches for the machine to take which will reach a valid accepting state. So, given any instance of a problem without this extra bit of information, the problem seems to be very difficult. However, with the extra information from the "proof," it is trivial to check to see if the instance will be accepted or not. Commonly, this method is used to determine if a problem lies within the NP complexity class, since if it can be easily verified to be valid given a polynomial sized proof string, then it must lie within NP. Thus, NP problems can be seen as problems with finding the existence of such a ”proof” within some proof-space.

For example, the Hamiltonian path decision problem lies within NP, since the easily verified proof of the existence of such a path would be the path itself. For any NP problem, the proof space will be known and provides a clean method for determining the solution to an instance through brute forcing all possible proofs for that instance of an NP problem and rejecting if none are found. With some effort, it is possible to see that such a proof-space will be finite but exponential based on the number of multivalued transitions. Obviously, testing every proof in the space is inefficient, but possible. With such an algorithm, if a proof exists, then the instance of the problem will eventually be accepted. If not then it is rejected after exhausting the proof-space. For the Hamiltonian path problem, given an instance of a graph G with n vertices, the proof-space would be every single path of length n. Once a Hamiltonian path is found, it is easy to show someone that the graph has such a path by simply providing them with the one which was found.

2.5 The Boolean SATISFIABILITY Problem

An essential decision problem to computational complexity theory is the Boolean SATISFIABILITY problem. This problem has the unique property of being NP-Complete, and was the first problem ever discovered to be so. In fact, all other NP-Complete problems are proved to be so through polynomial reduction from the satisifiability problem. The satifiability problem is as follows:

Boolean SATISFIABILITY Problem Given a Boolean expression written using NOT, the Boolean conjunctions AND and OR, a finite set of variables and appropriate parentheses, does there exist a Boolean assignment to the variables for which the Boolean expression will evaluate as 1? In logic language, does there exist an assignment of "TRUE" and "FALSE" to the variables for which the expression remains "TRUE?"

For the purposes of clarity, this thesis will refer to the Boolean SATIFIABILITY problem as the SATIFIABILITY problem from this point forward. Here is a simple example of what an instance for the SATISFIABILITY problem would look like (using logical notation):

$$I = (x_1 \vee x_2 \wedge (\neg x_1 \vee x_5) \vee (\neg(y \vee x_6))), \{x_1, x_2, x_5, x_6, y\}$$

In this example, there are a variety of valid values for the variables (x_1, x_2, x_5, x_6, y). Notice that the instance also includes the list of variables valid for that instance, as there could be variables included that are not incorporated in the Boolean expression. While, doing this would be useless, it is not outside the definition of the SATISFIABILITY problem.

It is fairly clear that this problem will lie within NP since the Boolean assignment of the variables is going to be polynomial in size compared to the instance, and can act to show that the system has a assignment which is accepted (i.e. a proof). However, the fact that it is complete in that complexity class is not as clear. The following theorem is by far the most important theorem in the field of computational complexity. It is the first non-trivial completeness proof and provides a base to determine if other problems lie within the NP-Complete complexity class.

Theorem 2.5.1 (Cook-Levin Theorem)**.** *The SATISFIABILITY problem is NP-Complete.*

Proof. This proof can be located in Garey and Johnson's book, "Computers and Intractability: A guide to NP-Completeness" [28]. □

To make it simpler to deal with the SATISFIABILITY problem, it is useful to use the set of subproblems where the Boolean expression is restricted to the conjunctive normal form with clauses of a specific length, k. These problems are known as k-SAT problems, and provide an infinite set of subproblems to SATISFIABILITY, which when combined together form the SATISFIABILITY problem space.

k-SAT k-SAT is a subset of the SATISFIABILITY problem, where the Boolean expression is written in the conjunctive normal formal with each clause having at most k elements. This means that the Boolean expression has the following form, where a_{ij} is a Boolean variable or a negation of a Boolean variable and it is possible for a variable to appear multiple times in any part of the expression.

$$(a_{11} \vee a_{12} \vee \ldots \vee a_{1k}) \wedge (a_{21} \vee a_{22} \vee \ldots \vee a_{2k}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee \ldots \vee a_{nk})$$

Note that $\bigcup_{k=0}^{\infty}$ k-SAT = SATISFIABILITY, as any SATISFIABILITY problem can be easily rewritten into the conjunctive normal form. In fact, it is possible to polynomially reduce any k-SAT problem to a 3-SAT problem making 3-SAT a small NP-Complete set. Note that, while 3-SAT is NP-Complete, 2-SAT lies within P and if it is believed that $P \neq NP$, then 2-SAT cannot be NP-Complete.

The following proof demonstrates how polynomial reducibility can be used to prove the NP-completeness of a decision problem.

Theorem 2.5.2. *3-SAT is NP-Complete*

Proof. To prove that 3-SAT is NP-Complete, we will take a generalized instance of SATISFIABILITY and map it to 3-SAT.

Given an instance of SATISFIABILITY written in conjunctive normal form, $I = \{C_0, C_1, \ldots, C_n\}, \{x_0, x_1, \ldots, x_m\}$, where C_i are the clauses and x_i are the variables, there will be 3 different types of clauses which will need to be converted to clauses of length 3. The clauses that are already of length 3 do not need to be converted.

1. If C_i is a clause of length 1. Then $C_i = x_j$ for some j. It is possible to replace C_i with a set of 4 new clauses which are satisfiable if and only if C_i is satisfiable by adding 2 new variables y_{i1}, y_{i2}.

$$C'_{i1} = (x_j \vee y_{i1} \vee y_{i2}), C'_{i2} = (x_j \vee \neg y_{i1} \vee y_{i2}), C'_{i3} = (x_j \vee y_{i1} \vee \neg y_{i2}), C'_{i4} = (x_j \vee \neg y_{i1} \vee \neg y_{i2})$$

2. If C_i is a clause of length 2. Then $C_i = (z_j \vee z_k)$ for some z_j, z_k representing the variables in normal or negated form. By adding a new variable y_i, it is possible to replace C_i with two clauses of length 3 which are satifiable if and only if C_i is satisfiable.

$$C'_{i1} = (z_k \vee z_k \vee y_i), C'_{i2} = (z_k \vee z_k \vee \neg y_i)$$

3. If C_i is a clause of length k, $k > 3$, $C_i = (z_1, z_2, \ldots, z_k)$. Then it is possible to create a set of k-3 new clauses of length 3 which are satisfiable if and only if C_i is satisfiable. First it is necessary to add $k-3$ new variables, $\{y_{i1}, y_{i2}, \ldots, y_{i,k-3}\}$ and replace C_i with the following set of clauses.

$$C'_{i1} = (z_1, z_2, y_{i1}), C'_{i,l} = (\neg y_{i,l}, z_{l+2}, y_{i,l+1}) \forall 1 < l < k-3, C'_{i,k-3} = (y_{i,k-3}, z_{k-1}, z_k)$$

With these conversions, we now have a 3-SAT problem which is satisfiable if and only if the original problem is satisfiable. However, it is necessary to show that this reduction is a polynomial reduction based on the size of the instance $|I| = n + m$. Let n_1 be the number of clauses of length 1, n_2 be the number of clauses of length 2, n_3 be the number of clauses of length 3, and n_k be the number of clauses of length $k > 3$. It is clear that $n_1 + n_2 + n_3 + n_4 = n$. Let n' be the new number of clauses and m' be the new number of variables.

Now for the n_1 clauses of length 1, there are 2 new variables and 3 extra clauses. So the 3-SAT reduction produces $n_1 * 4$ clauses and at most $n_1 * 3$ variables. Similarly for

n_2, there are now $n_2 * 2$ clauses and at most $n_2 * 3$ variables. For n_k the solution is a bit more complicated, so to make it clearer it is useful to use big-O notation. Let $K > 3$ be the size of the biggest clause in the instance. So, all other clauses will increase the size by less than the amount this clause increases the system. So, for $k > 3$, there are now $2k - 3$ variables and $k - 3$ clauses. Since K is the largest possible K, there are now at most $n_k * (2K - 3)$ variables and $n_k * (K - 3)$ clauses. It is easy to see that if $K > 3$, then the entire systems growth is bounded above by by $n * (2K - 3)$ variables and $n * (K - 3)$ clauses. Therefore the reduction is $O(n)$. Thus this is a polynomial reduction of SATISFIABILITY to 3-SAT and since SATISFIABILITY is NP-Complete, this implies that 3-SAT is NP-Complete. $\square$

Using 3-SAT, it becomes possible to prove that other problems are NP-Complete without having to use the complicated techniques developed to prove that SATISFIABILITY was NP-Complete. With 3-SAT as an established NP-Complete problem, any problem to which it polynomially reduces must also be an NP-Complete problem. This is due to the fact that any concatenation of two polynomial reductions produces yet another polynomial reduction. In fact, if a tree were to be drawn regarding the known NP-Complete problems, the root of the tree is the SATISFIABILITY problem with a majority of the problems coming off of the initial 3-SAT branch of the tree [28]. In the MULTISTABILITY chapter of this thesis, 3-SAT will be used to demonstrate the difficulty of this problem in the general case.

Another important complexity class related to NP is the coNP complexity class. From a decision problem viewpoint this is the inverse problem. So, for a given SATISFIABILITY problem, the related co-NP problem would be "does this SATISFIABILITY problem have no Boolean string such that it equals zero?" Unlike the NP problem where simple existence is needed to validate whether the problem is true or not, the coNP problem requires universality, i.e., all possible answers must return "reject," and thus fail to work.

2.6 PSPACE

One last complexity class completely rounds out this family of polynomial complexity classes: PSPACE. This is the class of all problems which are solvable using a polynomial number of resources, but an unlimited amount of time. Within this class, lie the classes P, NP, and coNP. While it initially seems that this problem would not be directly related to P, NP and coNP, there is a very fascinating connection between their primary complete problems. NP-Complete becomes a question of the existence of a proof, coNP-Complete is a question of the universality of a statement, and PSPACE is a question of universality for a given subset.

There is an interesting logical way to represent the complete classes for NP, coNP, and PSPACE using equivalent satisfiability problems. The NP-Complete class is represented by the SATISFIABILITY problem, given a Boolean expression $B, \exists \overline{X} \in \{0,1\}^n$ where $B(\overline{X}) = 1$. The coNP-Complete class by, given a Boolean expression $B, \forall \overline{X} \in \{0,1\}^n$ where $B(\overline{X}) = 1$. PSPACE-Complete class is represented by, given a Boolean expression $B, \exists \overline{Y_0} \in \{0,1\}^{n_0}, \forall \overline{X_0} \in \{0,1\}^{m_0}, \exists \overline{Y_1} \in \{0,1\}^{n_1}, \forall \overline{X_1} \in \{0,1\}^{m_1}, \ldots, \exists \overline{Y_k} \in \{0,1\}^{n_k}, \forall \overline{X_l} \in \{0,1\}^{m_l}$ where $B(\overline{X}, \overline{Y}) = 1, \overline{X} = \{\overline{X_0}, \ldots, \overline{X_l}\}, \overline{Y} = \{\overline{Y_0}, \ldots, \overline{Y_k}\}$. Thus, PSPACE-Complete problems end up being a combination of coNP-Complete and NP-Complete problems. Thus, naturally, $NP, coNP \subseteq PSPACE$ [28]. However, as discussed before, the strictness of the inclusion has not yet been proven.

2.7 Relationships between the classes

Figure 2.3 shows the current believed relationships between the classes of P, NP, coNP, PSPACE, PSPACE-Complete, NP-Complete, and coNP-Complete.

While the strictness of the inclusions within figure 2.3 have all not been proven, the figure assumes that $NP \neq P$, $PSPACE \neq NP$, $coNP \neq NP$. This is why the complete subclasses are shown as separate from the other subclasses for each complexity class. Eg.

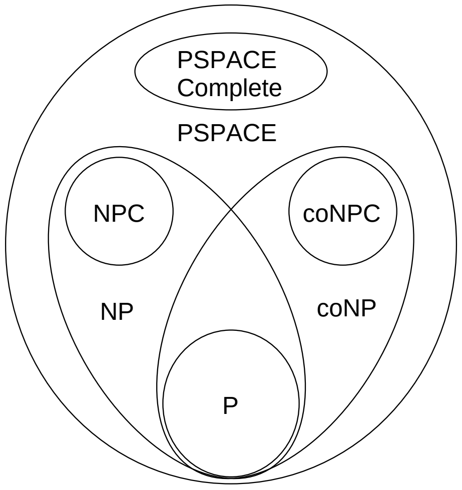


Figure 2.3: Believed set relations between complexity classes related to P, NP and PSPACE

NPC is shown as not being within or even intersecting P.

If it turns out that $P = NP$, then $coNP = NP$, because $coP = P$. Similarly, having $PSPACE = NP$ would imply $coNP = NP$, and collapse a significant portion of the hierarchy. It is believed amongst a good portion of computational complexity theorists that $P = NP \cap coNP$. However, there is no solid proof of this as of yet. Any results or development to refine this diagram to be more rigourous would be priceless for furthering the understanding of tractability and "efficient" complexity classes.

Chapter 3

Continuous Models of Genetic Regulatory Networks

3.1 Introduction

Stuart Kauffman developed the Boolean network model for genetic regulatory networks in the late 1960s and proceeded to demonstrate interesting ensemble properties of the discrete dynamics of the model. The simplicity of his model allowed for potential qualitative features for certain Boolean network dynamics to be teased out of the structure of the system [64, 20, 41, 42, 60, 66]. From these features, it became possible to intuit ensemble dynamic features of genetic regulatory networks to test empirically, assuming that the system dynamics can be reasonably represented by a discrete synchronous system.

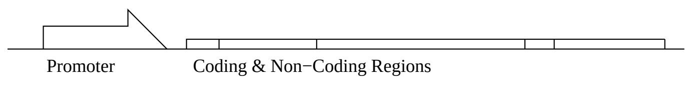
Unfortunately, even though Boolean networks can provide some intuition, genetic regulatory networks are fundamentally not discrete synchronous systems. Therefore, applying any qualitative behaviour about ensemble Boolean network dynamics to genetic regulatory dynamics must be considered carefully. Therein lies the biggest fundamental difficulty when creating any model, especially models of a genetic regulatory network. Creating a model requires finding a balance between model simplicity and conformity to real-world phenomena. I.e., keeping the model simple enough to derive rigourous conclusions from it and ensuring that the model is accurate enough to be representative of a real-world genetic network, especially if said model is to be used with any algorithms designed to determine qualitative properties of the real-world system like MULTISTABILITY or finding attractors.

To create a useful model of a genetic network, it is necessary to understand the biology of cells [3] and the biology of genetic regulatory networks [10]. Specifically, we need to know which features of the biology have the largest effects on the qualitative behaviour of the dynamics. These will be shown to be the gene expression and the activation and repression thereof. Once this is identified and understood, a model can be created using a few methods. A simplified set of chemical master equations can be written providing the expression, activation and repression. From these equations, either the Gillespie algorithm can be used to simulate the chemical dynamics [59], or a Mean Field Model approach can generate an associated system of ODEs [4]. Another method ignores the chemical master equations entirely and instead directly represents activation and repression with hill or Heaviside functions in a system of ODEs representing gene expression [32].

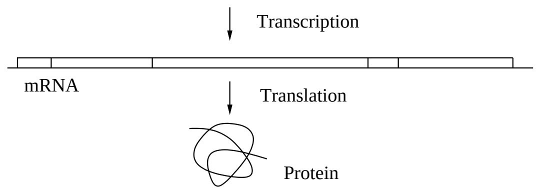
3.2 Biology of Genetic Regulatory Networks

The eventual growth, differentiation, and behaviour of a cell is strongly affected by its internal molecular chemistry and dynamics [80]. The interactions between various proteins, mRNA, DNA, and other organic molecules direct the cell to behave in a variety of ways. In fact, there is strong evidence that the eventual cell fate is almost entirely controlled by switches created by genetic interactions [38]. However, understanding and modeling all of the molecular interactions of a cell is an immense task due to the sheer variety of possible organic molecules and the exponential nature of their interactions.

Due to this complexity, it becomes necessary to study subsystems which can be easily modeled. One subsystem which is becoming popular for studying long-term dynamics of a cell is the interaction between genes and their constituent proteins. This is commonly modeled as a regulatory network where each gene has a given expression value based on their associated protein concentration, and those concentrations either repress or activate other genes [10]. This chapter will provide demonstrative examples by using the PU.1, GATA-1 example from the introduction. As a reminder of how this example works, PU.1 and GATA-1 are both genes that mutually repress each other's expression,



(a) A Gene



(b) mRna and Proteins

Figure 3.1: Transcription and Translation

while simultaneously being self-activating. What is meant by expression, repression and activation will be explained chemically further on in this chapter.

3.2.1 Gene Expression

In a cellular context, a gene is a strand of DNA containing a promoter and coding and non-coding sequences. This can be seen in figure 3.1a. When a gene is expressed or active, the coding and non-coding sequence are copied, using a specialized promoter region for that gene, into RNA strands. These RNA strands can theoretically affect many aspects of a cell's behaviour and function. In the case of the genetic regulatory network model, the only RNAs considered are those which are involved with the production of the associated proteins to the gene. These RNA molecules, known as messenger RNA or mRNA, interact with the available ribosomes and create proteins that are directly related to the gene which produced the mRNA. The mRNA and proteins can be seen in figure 3.1b.

So, a common way that a gene expresses its current "state" is by what are referred to as the processes of transcription and translation as shown in a simplified form in figure 3.1, where transcription is the production of an associated mRNA from a gene, and translation is the chemical reaction of mRNA with ribosomes to produce an associated protein to the gene. In chemical reaction notation, transcription and translation are as follows.

For transcription, there are a variety of intermediary chemical reactions, represented here by $\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_n$. However, in most cases it is perfectly valid to simplify as in (3.2), where the transition is marked from the first reaction to the final one and the intermediate reactions are ignored. $\mathrm{P_m}$ represents the promoter region of the genome at gene m which initiates the transcription.

$$\mathrm{RNA_p} + \mathrm{P_m} \rightarrow \mathrm{C}_1 \rightarrow \mathrm{C}_2 \rightarrow \ldots \rightarrow \mathrm{C}_n \rightarrow \mathrm{RNA_p} + \mathrm{P_m} + \mathrm{mRNA_m} \tag{3.1}$$

$$\mathrm{RNA_p} + \mathrm{P_m} \rightarrow \mathrm{RNA_p} + \mathrm{P_m} + \mathrm{mRNA_m} \tag{3.2}$$

Translation commonly occurs directly from the interaction of the mRNA molecule with the ribosomes to produce proteins. Chemical equation (3.3) shows how this works. The Ribosomes within the cell interact with the mRNA in such a fashion that they produce a protein copy of the information from the mRNA. In the chemical equation, $\mathrm{M_n}$ is the associated protein created for gene n, given the $\mathrm{mRNA_n}$ from gene n.

$$\mathrm{Ribo} + \mathrm{mRNA_n} \rightarrow \mathrm{Ribo} + \mathrm{mRNA_n} + \mathrm{M_n} \tag{3.3}$$

While these chemical reaction equations are important to a chemical definition of a genetic regulatory network, these reactions are not the only important reactions. Without decay or degradation of the mRNA and proteins, the system will very quickly use up all available resources permanently. So, it is easy to see that the degradation of the mRNA and proteins will happen at some constant rate $k \geq 0$. Thus, we can define the chemical reaction equations for the decay of the proteins and the mRNA. Note, the Ribosome decay is not considered simply because it is assumed that there are more than enough to maintain the reaction at all times, and the promoter region also doesn't decay since that would imply that the DNA itself was breaking apart.

$$\mathrm{mRNA_m} \rightarrow \emptyset \tag{3.4}$$

$$\mathrm{M_m} \rightarrow \emptyset \tag{3.5}$$

From the perspective of the PU.1, GATA-1 system example, the equations needed would be the equations for the expression of the PU.1 gene, as follows:

$$\mathrm{RNA_p} + \mathrm{P_{PU.1}} \rightarrow \mathrm{RNA_p} + \mathrm{P_{PU.1}} + \mathrm{mRNA_{PU.1}}$$

$$\mathrm{Ribo} + \mathrm{mRNA_{PU.1}} \rightarrow \mathrm{Ribo} + \mathrm{mRNA_{PU.1}} + \mathrm{M_n}$$

$$\mathrm{mRNA_{PU.1}} \rightarrow \emptyset$$

$$\mathrm{M_{PU.1}} \rightarrow \emptyset$$

The chemical reactions for GATA-1's expression are the same. If this were the entirety of the biochemical system, the equations would imply that $M_{\mathrm{PU.1}}$ (and $M_{\mathrm{GATA\text{-}1}}$), the value we consider more important for representing gene expression, would simply reach a equilibrium point based on the various reaction rates and the system would quickly settle into a chemical equilibrium state. Obviously, it is necessary to add more chemical interactions.

3.2.2 Multimerization

Another important interaction within a cell is multimerization. This is when two proteins combine together to form what is referred to as a multimer. Commonly, this occurs between protein molecules of the same design. When this occurs between two proteins of the same construction, they are referred to as homodimers. When there are three or more of the same construction, they are commonly referred to as homomultimers. On the rare occasion, two or more protein molecules of different constructions combine. Again, when this occurs for two non-related proteins, they are called heterodimers, and for three or more, heteromultimers.

Dimerization can be represented by the balanced chemical equation (3.6). Multimerization is the natural extension of this equation. ($\rightleftharpoons$ implies that the chemical reaction is bidirectional)

$$\mathrm{M_i} + \mathrm{M_j} \rightleftharpoons \mathrm{M_iM_j} \tag{3.6}$$

In most biochemical systems, it is believed that the dynamics could be heavily influenced by homodimers in order to handle the intrinsic noise from the biochemical reactions [11]. In the PU.1, GATA-1 example, since the switches behaviour is stable, it is reasonable to assume that the proteins combine into homodimers. While this has not been empirically verified for this specific example, it is not unreasonable to believe that this is true. Thus, the PU.1, GATA-1 system will have the following chemical reactions in it:

$$\mathrm{M_{PU.1}} + \mathrm{M_{PU.1}} \rightleftharpoons \mathrm{M_{PU.1}M_{PU.1}}$$

$$\mathrm{M_{GATA\text{-}1}} + \mathrm{M_{GATA\text{-}1}} \rightleftharpoons \mathrm{M_{GATA\text{-}1}M_{GATA\text{-}1}}$$

3.2.3 Activation and Repression

With the above chemical equations (3.2) - (3.6), it is possible to define the behaviour of the most simple of genetic regulatory networks. Specifically, those which express themselves through protein production and potentially have their proteins interact to form larger compounds in some fashion. While this is useful, not very much happens in the way of dynamics for this system of equations simply because the proteins generated do not interact with each other in any way.

Simply put, a system consisting of purely gene expression without some form of regulation has no interesting dynamics. A common intuitive way to implement gene-

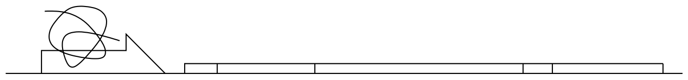


Figure 3.2: A protein binding to the promoter to cause repression

gene interaction is to have the expression of one gene repress or activate the expression of another gene. In fact, it seems that activation and repression are the most common methods that one gene's expression affects the dynamics [17]. In both circumstances the expression of one gene's proteins either directly, or through an intermediary multimer containing said protein binds to a promoter related to another gene. By binding to this site, the protein either reduces transcription of the associated gene's protein, a process known as repression, or chemically acts in a fashion similar to a catalyst by increasing the production of the protein, a process known as activation. By reducing the amount of mRNA created through transcription, it naturally reduces the equilibrium concentration of the related protein, thus decreasing expression. This can be seen in figure 3.2

Chemically, we can represent repression by having a given multimer D bind to the promoter $\mathrm{P_n}$ related to the transcription equation from before. Hence, the availability of the promoter $\mathrm{P_n}$, needed for transcription, decreases. As $\mathrm{P_n}$ decreases, the equilibrium of the transcriptional equation decreases, and thus the production of proteins for the associated gene similarly goes down. Chemically, this is represented as follows in equation (3.7) (Where D is the multimer and $\mathrm{P_n}$ is the repressed promoter region.) Since the dimer can attach or detach from the promoter region, this chemical reaction is bidirectional.

$$\mathrm{P_n} + \mathrm{D} \rightleftharpoons \mathrm{P_nD} \tag{3.7}$$

For activation, the promoter region will be bound to as above. However, instead of decreasing the production of the associated mRNA to 0, it will increase transcription by a value larger than the initial rate in the transcription equation. Thus, chemically, activation will require the above repression to produce the promoter-multimer pairing and the following chemical master equation, with a higher reaction rate than the original equation.

$$\mathrm{RNA_p + P_nD \rightarrow RNA_p + P_nD + mRNA_n} \tag{3.8}$$

In general, while it is possible for a single protein to bind to such a site, it seems as though the most important gene-gene interactions are caused by homomultimers, especially homodimers. Monomers are less able to deal with the intrinsic noise in the system [11] and heteromultimers are not commonly seen in experiment. Thus, even though a complete model would naturally include the full catalogue of monomers and heteromultimers, it is perfectly understandable when looking for multistable states to simplify the genetic regulatory network model by only looking at repression and activation caused by homodimers. However, any predicted result by a simplified model would need to be tested in a lab to ensure that the results are not complicated by any unexpected interactions. Thus, for the purpose of simplicity, this thesis will view all repression and activation as being caused solely by homomultimers, specifically homodimers.

So, in our PU.1, GATA-1 example, since PU.1 represses GATA-1 and vice versa, and this thesis assumes the repression is caused by the homodimers, the following chemical equations are added to the system to represent the mutual repression of these two genes.

$$\mathrm{P_{PU.1} + M_{GATA\text{-}1}M_{GATA\text{-}1} \rightleftharpoons P_{PU.1}M_{GATA\text{-}1}M_{GATA\text{-}1}}$$

$$\mathrm{P_{GATA\text{-}1} + M_{PU.1}M_{PU.1} \rightleftharpoons P_{GATA\text{-}1}M_{PU.1}M_{PU.1}}$$

As well, since PU.1, and GATA-1 are self-activating, the following equations must also be within the set of chemical reactions. Note that for them to be self-activating, the rate of reaction, which is not currently included with the equations, is higher for these than for the regular transcription.

$$\mathrm{P_{GATA\text{-}1} + M_{GATA\text{-}1}M_{GATA\text{-}1} \rightleftharpoons P_{GATA\text{-}1}M_{GATA\text{-}1}M_{GATA\text{-}1}}$$

$$\mathrm{RNA_p + P_{GATA\text{-}1}M_{GATA\text{-}1}M_{GATA\text{-}1} \rightarrow RNA_p + P_nM_{GATA\text{-}1}M_{GATA\text{-}1} + mRNA_{GATA\text{-}1}}$$

$$\mathrm{P_{PU.1} + M_{PU.1}M_{PU.1} \rightleftharpoons P_{PU.1}M_{PU.1}M_{PU.1}}$$

$$\mathrm{RNA_p + P_{PU.1}M_{PU.1}M_{PU.1} \rightarrow RNA_p + P_nM_{PU.1}M_{PU.1} + mRNA_{PU.1}}$$

3.3 ODE Models of Genetic Regulatory Networks

As discussed in the introduction, a major problem with relating a discrete model such as a Boolean network to its associated real-world system is that a Boolean network is discrete, synchronous and not noisy. Probabilistic Boolean network models have been developed by Shmulevich, but the problems regarding the discrete nature of the system still remain[63]. So, although results from the Boolean network model can be robust in some circumstances, these properties make some conclusions based on this model to be fragile [13]. Naturally, the Boolean network's dynamics are heavily affected by those three properties and any conclusions thereof may not necessarily apply directly to its associated real-world system. As expected, recent papers seem to show that noise and asynchronicity can be important factors with regards to certain important dynamics regarding cell regulation and the cell cycle [68, 26]. As well, the continuous nature of the real-world system may lead to fixed points which are not stable, but in a discrete system would appear to be so.

There are two primary methodologies used to develop a continuous model of a genetic regulatory network. One method is to start from the discrete model of the Boolean network and convert it to an equivalent continuous system. This was done first by Leon Glass, when he created piecewise linear ordinary differential equations (PLODEs) [32], and has been used in many variations since to study the mathematical behaviour of various systems [31]. While ODE numerical simulation can be computationally intensive for large systems, an interesting new technique to study its overall dynamics is qualitative simulation of piecewise linear ODEs [19]. This simulation uses the linear nature of ODEs outside of transitions for PLODEs to simulate the dynamics more rapidly. While this thesis will not go into deep technical detail on these models as they do not relate to the fundamental topic of it, piecewise linear ODEs and qualitative simulations of such are mathematically interesting regardless.

Another method is to start with an extensive, commonly intractable, chemical and physical model of the system and gradually simplify features of the system until the model becomes tractable. At its most sophisticated, the individual chemical reactions are computationally simulated using the Gillespie algorithm or some variation thereof [59]. While these methods allow for reasonably efficient numerical simulation, they are difficult to analyze rigourously. In a less sophisticated vein, non-stochastic models can be useful, commonly for very small toy systems like the repressilator or toggle switch [74]. However, until recently, there was not a clear and simple way to do this without the system being equivalent to some variation on Glass's piecewise linear ODEs.

However, in 2006, Andrecut published a paper describing how to use a Mean Field Model approach with the chemical equations for the system to map a given genetic regulatory network whose dynamics are primarily based on activation and repression to a system of ODEs [4]. This is the primary method this paper will use in its analysis of the MULTISTABILITY problem as this is derived directly from the chemical equations and remains simple enough to draw some important analytic results.

3.3.1 Piecewise Linear ODEs

Shortly after Kauffman developed the Boolean network model of genetic regulatory networks, Glass took the general idea of switch based interaction and adapted it to a continuous system of differential equations known as piecewise linear differential equations. To do this, he would use hill functions or threshold based Heaviside functions within the system of differential equations, whereby when a certain combination of concentrations of proteins were above or below defined thresholds the Heaviside function would either be "on" or "off". This had the effect of simulating within a continuous, asynchronous environment something resembling a Boolean network model [32].

For a genetic regulatory network with no compartmentalization, a system of differential equations is laid out as follows. First, for each gene n, a level of "expression" is defined as $x_n \in \mathbb{R}^+$. Given these variables, a system of ODEs is constructed as follows.

$$\frac{dx_n}{dt} = M(\overline{X}) - x_n$$

Where,

$$\overline{X} = \{x_1, x_2, \ldots, x_n\}$$

Where M is either a hill function or Heaviside function based on a Boolean function such that when the appropriate genes reach a certain threshold of expression (either high or low) the function shifts from low to high or vice versa. For a simple system based entirely on repression and activation, the M function would simply be a measure of whether activation is higher than repression for any given gene. In most situations, it is assumed that if the only effect any genes have on another gene x is repression, then M is high unless one of those repressive genes is being expressed.

So, figure 3.3 represents a mutually repressive two gene system. For this system, if gene A is high, then gene B is repressed and vice versa. This was the original biological switch using genetic expression and repression [14]. In systems biology this is commonly referred to as a toggle switch and is seen regularly within known real-world genetic regulatory networks. Note that this is the PU.1, GATA-1 system without self-activation. By

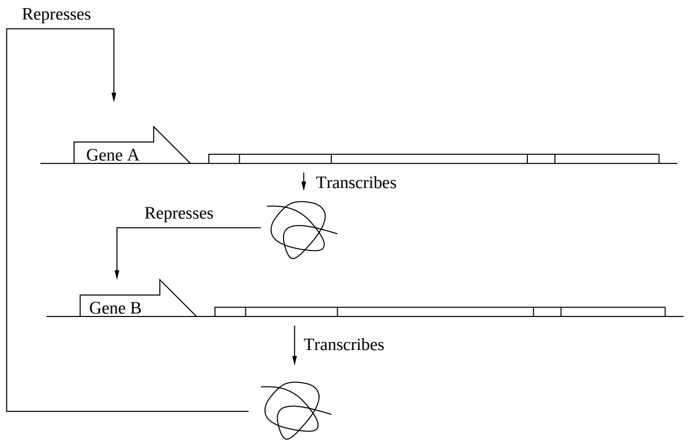


Figure 3.3: A Simple Genetic Toggle Switch

taking Glass' piecewise linear ODEs using hill functions, this system would be represented as follows.

$$\frac{dx_A}{dt} = 5\frac{1}{1+x_B^2} - x_A, \frac{dx_B}{dt} = 5\frac{1}{1+x_A^2} - x_B \tag{3.9}$$

This system of ODEs functions, as would be expected, has three fixed points, two of which are stable. This can be seen in the separatrix in figure 3.4. As such, it behaves like a switch with two stable fixed states, High/Low or Low/High, and a unstable point where the two expressions are exactly equal. The PU.1, GATA-1 system is generally seen as tristable experimentally, where the stability is dependent on the rates of the various chemical reactions in the system. For some reaction rates, the self-activation can provide a very stable fixed point where the two are exactly equal, and the mutual repression creates the other two fixed points, similar to to this system's behaviour.

Similarly, figure 3.5 shows what is known as a repressilator, a cycle of three repressive genes. Mathematically, this system's set of ODEs would look very similar to the toggle switch. In this system, there is only one fixed point, and it is unstable. This fixed point is when all three genes are equally expressed. It is a very simple example of how genes can be wired such that their behaviour is periodic.

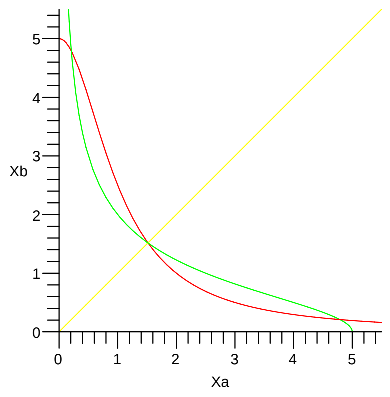


Figure 3.4: The separatrix of the toggle switch PLODE system given by the system of ODES in equation (3.9)

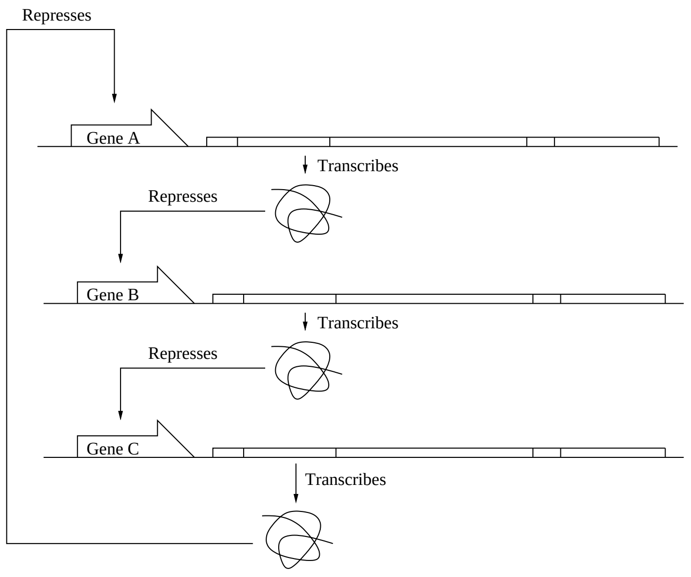


Figure 3.5: A Simple Genetic Repressilator

Both the toggle switch and repressilator will be used in the MULTISTABILITY chapter to show that MULTISTABILITY is NP-Complete.

In 2000, Elowitz et al. genetically modified a strain of e-coli to fluoresce in different colours when certain genes were activated. They then also modified the genes to be mutually repressive. In their paper, they created a repressilator and showed that it oscillated in a fashion similar to that predicted by Glass' model using hill functions [25]. In a similar paper by Gardner et al., a toggle switch was created in vitro which was able to chemically flip by over-expressing the associated genes [27]. Both papers strongly suggested that noiseless ODE models of genetic regulatory networks may be an incredibly powerful resource when dealing with subnetworks and determining important dynamical properties, and be especially useful for determining properties related to MULTISTABILITY. They even imply that even though it is known that noise can significantly change the behaviour at times, in general the expected behaviour is reasonably accurate.

3.3.2 The Mean Field Model

Given the chemical master equations (3.2) - (3.8), it is possible to use statistical averaging methods to define the continuous dynamics as would be expected from any given system. Here are the important chemical equations again with rates of reaction. As discussed before, this thesis will only consider repression and activation by dimers for the purposes of simplicity. For more generalized systems, see Andrecut's 2006 paper [4]. All of these formulas are as described from the previous section, with the respective rates placed above the reaction direction.

Transcription and mRNA decay

$$\mathrm{RNAp} + \mathrm{P_n} \xrightarrow{k_I^n} \mathrm{RNAp} + \mathrm{P_n} + \mathrm{mRNA_n}$$

$$\mathrm{mRNA_n} \xrightarrow{\overline{k}_I^n} \emptyset$$

Translation and protein decay

$$\mathrm{Ribo} + \mathrm{mRNA_n} \xrightarrow{k^n_{II}} \mathrm{Ribo} + \mathrm{mRNA_n} + \mathrm{M_n}$$

$$\mathrm{M_n} \xrightarrow{\overline{k}^n_{II}} \emptyset$$

Dimerization

$$\alpha_{ij} : \mathrm{M_i} + \mathrm{M_j} \xleftarrow{\overline{k}_{ij}}\xrightarrow{k_{ij}} \mathrm{M_iM_j}$$

Repression

$$\lambda^n_{ij}\alpha_{ij} : \mathrm{P_n} + \mathrm{M_iM_j} \xleftarrow{\overline{k}^n_{ij}}\xrightarrow{k^n_{ij}} \mathrm{P_nM_iM_j}$$

Activation

$$\mu^n_{ij}\lambda^n_{ij}\alpha_{ij} : \mathrm{RNAp} + \mathrm{P_nM_iM_j} \xrightarrow{k^n_{ij}} \mathrm{RNAp} + \mathrm{P_nM_iM_j} + \mathrm{mRNA_n}$$

Here, $\alpha_{ij}, \lambda^n_{ij}, \mu^n_{ij} \in \{0, 1\}$ depending on whether or not that reaction exists within the associated network. As noted, for a protein to activate another gene, it must first bind to the associated promoter and thus essentially act as a “repressor”. This is because the binding fundamentally represses the basic basal rate, but activates a potentially higher rate of protein production.

Note that there is only one promoter for each gene within the system at any time, and the promoter can lie in a list of possible states. One state all promoters have is the base state where nothing is attached to the promoter. If any protein or protein complex can attach to a promoter given the system of chemical equations, then the state where that promoter is attached to that protein complex is a possible state. All promoters have a certain possibility of lying in each of these states, and when they are in said state the entire system will behave accordingly.

In chemical equilibrium, the concentrations will be statistically as described in (3.10),(3.11) for the dimerization, and repression rates. Here $[X]$ represents the concentration of chemical X in the system. It is easy to see that when a system is in equilibrium, any dynamical representation of the system will be in a fixed state.

$$k_{ij}^{n}[\mathrm{P_n}][\mathrm{M_iM_j}] = \overline{k}_{ij}^{n}[\mathrm{P_nM_iM_j}] \tag{3.10}$$

$$k_{ij}[\mathrm{M_i}][\mathrm{M_j}] = \overline{k}_{ij}[\mathrm{M_iM_j}] \tag{3.11}$$

This thesis will only examine systems where dimers have an effect. So, as discussed, a gene's promoter can lie in one of the following states: $\{\mathrm{P_n}, \mathrm{P_nM_iM_j}\}$, i.e., either the promoter is free, or it is bound to a dimer, and there is some probability of it being in either of these states. From the equations, it is possible to determine the probability of a promoter lying in either of these states. The entire system potentially behaves with completely different dynamics based on the state of each promoter. By taking the field of all possible dynamics given all promoter states, and then taking the mean of said field given the probabilities of each state, it becomes possible to create a mean field estimate as to how the various proteins and mRNAs will be produced or decay on average at any time. Thus, it becomes possible to roughly model the dynamics of the genetic system.

Let x_i be the reduced concentration variable for the transcription factor associated to i. Then,

$$\frac{[\mathrm{P_nM_iM_j}]}{[\mathrm{P_n}]} = \frac{k_{ij}^{n}}{\overline{k}_{ij}^{n}}[\mathrm{M_iM_j}] = \frac{k_{ij}^{n}}{\overline{k}_{ij}^{n}}\frac{k_{ij}}{\overline{k}_{ij}}x_i x_j = (b_{ij} + b_{ji})x_i x_j = 2b_{ij}x_i x_j,$$

where b_{ij} represents the influence of the ij multimer on the system. As the order of i, j is not important, it is just assumed $b_{ij} = b_{ji}$. This is done to make later equations more clear. Now from this equation it is possible to determine the probability of a promoter $\mathrm{P_n}$ being in a free state.

$$\pi_0^n = \frac{1}{1 + \sum_{ij} \lambda_{ij}\alpha_{ij}b_{ij}x_ix_j}.$$

This is where the denominator is the sum of all of the possible states of the promoter with the b_{ij} influencing the possibility of being in the repressive state rather than the free state. Similarly, the probability of $\mathrm{P_n}$ being in the state where it is attached to a dimer $\mathrm{M_iM_j}$.

$$\pi_{ij}^n = \frac{\lambda_{ij}\alpha_{ij}b_{ij}x_ix_j}{1 + \sum_{ij} \lambda_{ij}\alpha_{ij}b_{ij}x_ix_j}.$$

If the system were expanded to use all possible types of multimers, the various probabilities for the states of a given promoter are generated in a similar fashion. Andrecut's paper on this model does this [4].

Now that the probabilities that the system's promoters are in a given state is known, it is possible to approximate the transcription of $\mathrm{mRNA_n}$ by taking the mean of the possible dynamics given those probabilities. This is accomplished by noting that when the system is in a steady state, the $\mathrm{mRNA_n}$ transcription will be equal to the $\mathrm{mRNA_n}$ decay. So if and only if this is true are the differential equations for the dynamics equal to 0. Using this fact, the chemical dynamics can be approximated. Let y_n be the mRNA concentration of the n^{th} gene. Then,

$$\frac{k_I^n + \sum_{ij} k_{ij}^n\mu_{ij}^n\lambda_{ij}^n\alpha_{ij}b_{ij}^n x_ix_j}{1 + \sum_{ij} \lambda_{ij}\alpha_{ij}b_{ij}^n x_ix_j} = \overline{k}_I^n y_n,$$

$$\eta_n \frac{1 + \sum_{ij} \lambda_{ij}\alpha_{ij}d_{ij}^n b_{ij}^n x_ix_j}{1 + \sum_{ij} \lambda_{ij}\alpha_{ij}b_{ij}^n x_ix_j} - y_n = 0,$$

where $\eta_n = \frac{k_I^n}{\overline{k}_I^n}, d_{ij}^n = \frac{k_{ij}^n}{\overline{k}_I^n}$. Both of these can be viewed as the "promoter strength" given that current state of the promoter since it provides how much the promoter overcomes the natural decay rate of the system. With this equation, it is possible to define the set of nonlinear differential equations which will approximate the transcription of gene n:

$$\frac{dy_n}{dt} = \eta_n \frac{1 + \sum_{ij} \lambda_{ij}\alpha_{ij}d_{ij}^n b_{ij}^n x_i x_j}{1 + \sum_{ij} \lambda_{ij}\alpha_{ij} b_{ij}^n x_i x_j} - y_n.$$

Similarly, by recognizing that in a steady state, the concentrations $[mRNA_n]$ and $[M_n]$ need to be in balance according we see that $k_{II}^n[mRNA_n] = \overline{k}_{II}^n[M_n]$. Thus, letting $\theta_n = \frac{k_{II}^n}{\overline{k}_{II}^n}$ we have the following set of differential equations which approximate the translation.

$$\frac{dx_n}{dt} = \theta_n(y_n - x_n), n = 1, \ldots, n$$

With these two sets of differential equations, it becomes possible to describe statistically the mean dynamics of the simplified genetic regulatory network, given no intrinsic or extrinsic noise. As stated, with very little work this model can be extended to use any combination of multimers and even include more chemical reactions. However, for the purpose of studying the MULTISTABILITY problem it is acceptable to use this simplified version.

A clear benefit to using the Mean Field Model compared to most other approaches is that this model comes directly from the chemical master equations for the system. So, unlike results with Glass' PLODES, any analytical results the Mean Field Model provides relate directly to a specific chemical system. In fact, the hill function-based PLODEs of Leon Glass's work can be easily simulated as a subconstruct within this system by choosing appropriate variables for the translation and repression strengths, and the Heaviside function-based PLODEs can be simulated by allowing for extremely large multimers which are rapidly created.

Due to its direct connection to the physical system, and the relative ease of its construction, the Mean Field Model will be the primary model used to study the problem of MULTISTABILITY within genetic regulatory networks within the following chapter.

With regards to the PU.1, GATA-1 example, the chemical equations given in the previous section would simply have the associated empirically determined rates assigned to them, and then the average statistical behaviour of the system could be simulated with this model.

3.4 Other Continuous Models of Genetic Regulatory Networks

As discussed in the previous section, both the PLODE and Mean Field Model approach produce a very useful mathematical model which can be studied analytically to study interesting aspects of the behaviour of a genetic regulatory network. In both cases, the models can be extended to include spatial compartmentalization and delays as necessary to make them more empirically sound [32]. However, notably missing from both models is the property of noise, both intrinsic noise from the timings of the chemical reactions and extrinsic noise from outside the system. The noise can obviously be incorporated in the differential equation models by modifying them to be appropriate systems of stochastic differential equations. Yet, this requires the addition of a layer of artificiality for some of the noise which makes connecting computational results with real world effects difficult.

To avoid the artificiality of "adding noise" to the differential equation models, a method has been developed which uses the Gillespie algorithm to simulate the chemical process probabilistically [29, 30]. In this model, the chemical equations are mapped to associated chemical master equations similar to the Mean Field Model. Intrinsic noise is incorporated by the Monte Carlo methods used to simulate the chemical equation, and outside noise can be easily added in a fashion which is clearly connected to real world phenomena [59].

For the purposes of understanding the MULTISTABILITY problem, this lack of noise is not likely to have a huge effect. If the amount of noise is kept minimal, the stable fixed points will, by definition, remain fixed in the associated real world network. However, if the model is used to simulate the real world dynamics, or determine possible limit cycles, the noise can have a very important effect with regards to the dynamics [26], especially when it comes to a cell's differentiation pathway. In fact, some recent work has shown that noise may end up being very important for some of the cellular processes observed in the laboratory, like the cell cycle [68].

Chapter 4

The MULTISTABILITY Problem

4.1 Introduction

In the study of genetic regulatory networks, the problem of finding "genetic switches" is a matter of determining which subsystems have stable or semi-stable fixed states within the genetic regulatory network. This problem, known as the MULTISTABILITY problem, will be defined as follows:

MULTISTABILITY (non-rigourous) Given an instance of a genetic regulatory network (GRN) (or subnetwork), does there exist more than one attractor which is fixed and stable for that network?

In this general form, this is not a rigourously defined mathematical or computational problem. The model is not clearly defined, and neither are the definitions of what fixed or stable means. As well, a given genetic regulatory network alone may not entirely encompass the full dynamics. This is due to the fact that the real world genetic network exists in a noisy environment with many external factors not incorporated within this definition, and the system itself may include other external chemical and quantum effects.

However, it is possible to rigourously define this biological problem given a specific model of genetic regulatory networks. This thesis will use the model developed from the previous chapter, the Mean Field Model. It is assumed that extrinsic noise is minimized, and the dynamics are affected entirely by chemical activation and repression with homo and heteromultimers. With these assumptions, the problem is rigourously defined using Andrecut's Mean Field Model of genetic regulatory networks based on the chemical master equations. MULTISTABILITY, defined with the most general version of the

Mean Field Model, is defined as follows:

MULTISTABILITY (mean field model) Given a system of ODEs based on the Mean Field Model [4]:

$$x_n, y_n \in \mathbb{R}^+$$

$$\frac{dy_n}{dt} = \eta_x \frac{1 + \sum_i \gamma_i^n \beta_i^n c_i^n a_i^n x_i + \sum_{i,j} \mu_{i,j}^n \lambda_{i,j}^n \alpha_{i,j} d_i^n b_i^n x_i x_j + \cdots}{1 + \sum_i \beta_i^n a_i^n x_i + \sum_{i,j} \lambda_{i,j}^n \alpha_{i,j} b_i^n x_i x_j + \cdots} - y_n$$

$$\frac{dx_n}{dt} = \theta_n (y_n - x_n)$$

The system is MULTISTABLE if and only if there exists more than one stable fixed point for this system of ODEs. The related MULTISTABILITY function problem is solved if all stable fixed states can be provided.

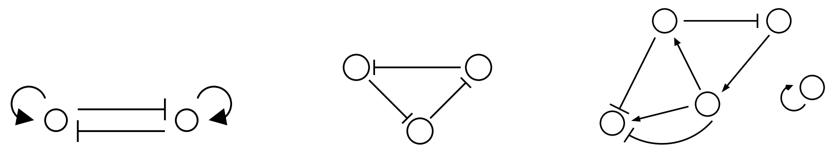
With the above definition of MULTISTABILITY, there remains an issue of defining a given GRN and choosing all of the appropriate constants. Naturally, if a complete human genome network were to be instantiated within this system, the definition of a single instance would be massive and intractable. The human genome has more than 20,000 genes, and including all possible heteromultimers of their proteins and all interactions would exponentially increase the size of the system. Thus, it is necessary to simplify the system to a set of base elements while maintaining most of the important functional dynamics of the original system. As discussed in a previous chapter on genetic regulatory networks, some biologists believe that a majority of the dynamics of GRNs are influenced entirely by either activation or repression of one gene by another, commonly by homomultimers. If restricted to this subset of MFMs, the problem of MULTISTABILITY becomes considerably more tractable and perhaps even efficiently solvable. As such, this thesis develops this idea of activation-repression systems.

4.2 Activation-Repression Systems

The Mean Field Model provides a way to study the dynamics of genetic regulatory networks. However, as seen in the introduction, it quickly becomes intractable to even define the system completely if all real world interactions are considered. Yet, if the dynamics are more or less dependent on the homomultimer, or even homodimer activation and repression, the Mean Field Model becomes considerably less complex. Unfortunately, the calculation of stable fixed points remains an elusive problem if the model remains in the form of a system of ODEs. This section will construct a related combinatorial object based on a pair of directed graphs, called a activation-repression system, which has a direct connection to the limited subset of Mean Field Models of genetic regulatory networks discussed above. This object distills the system of ODEs into a form which is simple and allows for a property called an A-R numbering to be defined. This property is believed to find all important stable fixed states.

Activation-Repression System An activation-repression system (A-R system) is a pair of digraphs on a given set of nodes N, and is represented by (N, A, R) where A and R are sets of directed edges on N.

A few examples of these A-R systems are presented in figure 4.1. As can be seen in the figure, the two digraphs are drawn on the same nodes with the edges being marked to indicate that they are from the A (activation) set or the R (repression) set of edges. In this thesis, the methodology will be as follows. The edge ends are changed based on which set they are from. If they are from the A set, they have an arrow at their tip. If they are from the R set, they have a flat line. For a given genetic regulatory network where dynamics are determined entirely by activation and repression reactions, the mapping to an A-R system is clear. Albeit, the rates of activation and repression are lost by this mapping. So, this is a forgetful map. If it is necessary to include rates,



(a) Toggle Switch with Self-Activation

(b) Repressilator

(c) Random A-R System

Figure 4.1: Examples of A-R Systems

a natural extension of this definition would have the edges weighted with their reaction rates, this extension will not be explored in this thesis.

In summary, to map a MFM to an A-R system, the genes which activate other genes are connected by directed edges within the A digraph, and genes which repress other genes are represented by directed edges in the R digraph. More complicated chemical interactions, such as those involving heteromultimers or other chemical compounds, will be ignored by this system. This should not be a significant problem, since it seems that most important genetic regulatory behaviour is affected by these direct interactions.

So, referring to the mutually repressive system of PU.1 and GATA-1, figure 4.1a would be the graphical representation of that system, where one node is labelled PU.1, and the other is labelled GATA-1. The looping arrow points back at itself to mark the self-activating nature of each node, while the flat-topped edges point at each other to represent the mutual repression.

With this graphical object representing genetic regulatory networks' dynamic interconnections, it is possible to map the problem of determining stable fixed states to a related combinatorial numbering problem.

A-R k-Numbering Given an A-R system S, and a $k \geq 2$, an A-R k-numbering $F : N \to \mathbb{Q}^+$ is a mapping such that the following condition is satisfied:

$$F : N \to \mathbb{Q}^+, F(n) = \begin{cases} 0 & \text{if } r(n) > 0 \text{ and } a(n) = 0 \\ 1 & \text{if } r(n) = a(n) = 0 \\ \frac{2*a(n)}{a(n)+r(n)} & \text{otherwise} \end{cases}$$

Where the functions a, r are defined as follows:

1.
$$\text{let } a : N \to \mathbb{Q}^+, a(n) = \sum_{m \to n \in A} F(m)^k$$

2.
$$\text{let } r : N \to \mathbb{Q}^+, r(n) = \sum_{p \to n \in R} F(p)^k$$

For ease of notation, it will be assumed that k always equals 2 whenever the above is referred to as an A-R numbering (without referencing the k).

This numbering is a simple way to represent a list of fixed states from an instance of the Mean Field Model for a given network with a large enough basal rate when all activations and repressions have very similar strengths and are based on homomultimers of size $k \geq 2$. As discussed earlier, without simplifying the system the problem of determining stable fixed points is obviously intractable. This is very important as it shows that if it is possible to find such a numbering for a given A-R system efficiently, then it is possible to determine fixed states for a large subset of genetic regulatory networks and partially answer the MULTISTABILITY problem. However, if this problem is provably intractable, then the MULTISTABILITY problem is guaranteed to be intractable or perhaps even uncomputable.

While MULTISTABILITY is concerned with the existence of multiple stable fixed points. It is true that given a MFM which matches the numbering, there will exist at least one fixed point with all values x_i are positive. However, this fixed point may be unstable.

Theorem 4.2.1. *Given a mean field model of a genetic regulatory network, where transcription is instantaneous, and all interactions are dimer based (i.e. $k = 2$), there is at least one fixed point $X = x_0, x_1, \ldots, x_n$ with all values $x_i > 0$.*

Proof. Since, transcription is viewed as instantaneous, we have the following vector field G in $\mathbb{R}^n$ (where $\overline{X} = \{x_1, x_2, \ldots, x_n\}$, and p_i are rational functions of even degree):

$$\frac{dx_i}{dt} = p_i(\overline{X}) - x_i, i = 0, 1, \ldots, n$$

Where $p_i(\overline{X}) > 0$ for all i and all values of $\overline{X}$, since the degree of the rational function p_i will always be even due to the restriction to dimer-based interactions. Consider the hyperplane on the axis $x_1 = 0$. The normal to this hyperplane is $N = (1, 0, \ldots, 0)$. For the dot product, we have $G \cdot N = p_1(\overline{X}) > 0$. This is true for all other axes. Therefore, the vector field points towards the positive orthant.

Create a large hypersphere, centered at the origin with radius r (i.e. $\sum_i x_i^2 = r^2$) in the usual form. The outward unit normal to the surface of this sphere will be $S = (x_1, x_2, \ldots, x_n)/r$. On the surface of this sphere, we have $G \cdot S \leq \sum_i p_i(\overline{X}) - \sum_i x_i \leq p_i(\overline{X}) - r$. So, if it is possible to show that there exists an r such that $r > \sum_i p_i(\overline{X})$, then the vector field points into the volume surrounded by the sphere. There exists such an r since the degree of the rational functions p_i are less than one for all i. Therefore, for large enough r, $r > p_i(\overline{X})$, for all i.

This surface has topological type of a hypersphere, and the vector field points inward to the volume encapsulated by the hypersphere and the axes. Therefore, the vector field on this surface has degree 1.

Therefore there must exist a zero inside the surface, and thus a fixed point. $\square$

To demonstrate how the A-R numbering can catch the fixed states for a system, one must simply look at figure 4.1a. In this figure there are two nodes which mutually repress each other while simultaneously being self-activated. Let the node on the left be a and the node on the right be b. The numberings show quickly why this is called a toggle switch. $F(a) = 2$, $F(b) = 0$ is one possible numbering. Similarly $F(a) = 0$, $F(b) = 2$ is also a valid numbering. However, since this system is self-activating, the numbering $F(a) = F(b) = 1$ is also valid since $F(a) = \frac{2F(a)}{F(a)+F(b)} = \frac{2F(a)}{F(a)+F(a)} = 1 = F(b)$. If the switch had no self-activation, then the only possible numberings would be $F(a) = 1, F(b) = 0$ and $F(a) = 0, F(b) = 1$. From a biological perspective this is encouraging, as various biological experiments have shown many of the differentiation switches within a cell are essentially self-activating toggle switches with exactly these three fixed states. It is theorized that these switches act as controllers for most observed cell behaviour. Specifically that they act as the originator of cellular differentiation. The fact that these three numberings match all possible stable fixed states for the self-activated toggle switch implies that the A-R system match the observed systems and provide insights into that behaviour.

This numbering provides us with most fixed states for a given GRN if the basal rate, η, is sufficiently large. However, the proof does not imply that these are stable fixed states for all possible rates. In fact, in certain cases they are not stable for certain parameters. For example, given the PU.1, GATA-1 self-activating mutually repressive system from above, if it has a low decay rate, the central fixed point $F(a) = F(b) = 1$ is an unstable fixed point. However, with a high decay rate and other certain parameters, there can become a small island of stability. What is particularly interesting, is while this doesn't have stable fixed states for all systems, it does give possible stable fixed states for all systems, and as the following example shows, when it misses a fixed state, that fixed state is not stable.

The repressilator in figure 4.1b has the property that there is no valid numbering. This is apparent from the fact that all nodes have the binary value of 1 or 0, because there is no activation and only repression. Also, the values are such that if $a \rightarrow b$, then $F(b) = \neg F(a)$. So writing out the system you have for $a \rightarrow b \rightarrow c \rightarrow a$, $F(b) = \neg F(a) =$

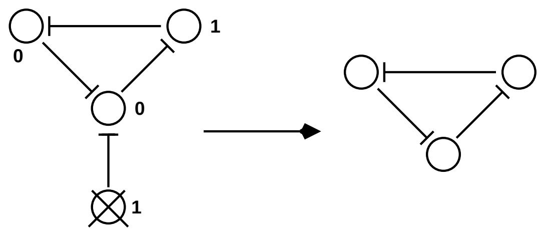


Figure 4.2: System w/ Numbering to Subsystem w/o Numbering

$F(c) = \neg F(a)$, thus $F(a) = \neg F(a)$, a contradiction. Hence, the repressilator has no valid numberings. In the equivalent genetic regulatory network, it is known that the repressilator has no stable fixed states, so this numbering again matches the observed behaviour. However, in the associated Mean Field Model system of ODEs there is a state when all three concentrations are equal, but this fixed state is unstable.

These two examples lead to Conjecture 4.2.2. It seems that while this numbering can sometimes catch potentially non-stable fixed states, it does catch all of the stable ones that do exist in the graph, and misses some of the non-stable ones. In the following section, the numbering will be adapted so it will provide only stable fixed states to a subset of A-R systems known as B-R systems. From this subset of problems it will be possible to show a lower bound to the computational difficulty of solving arbitrary A-R numberings and thus the MULTISTABILITY problem..

Conjecture 4.2.2. *All stable fixed states for the MFM have an associated possible numbering for the graph, independent of the chemical reaction rates.*

Definition A subsystem S of a given A-R system is an A-R system (N', A', R') such that $N' \subset N$ and $A' = \{(a_1, a_2) \in A | a_1, a_2 \in N'\}, R' = \{(r_1, r_2) \in |r_1, r_2 \in N'\}$. A subsystem is defined by the subset of nodes that it contains, i.e., $S \subset N$ is a way to represent a subsystem.

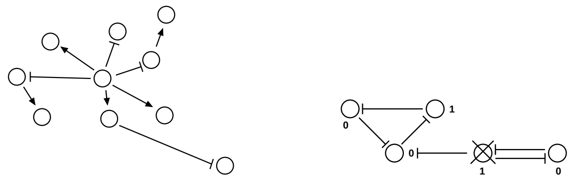
Obviously, having a valid numbering for any given subsystem $S' \subset N$ where $S' \neq N$ does not imply there exists a valid numbering for the main system. Similarly, if there exists a valid numbering to a given A-R system S, it does not imply that there will exist a valid numbering for any given subsystem, as is seen in figure 4.2. The full system on the left has a valid numbering (provided with the figure), however when the marked node is removed from the system to produce the system on the left, the subsystem becomes the repressilator, which, as discussed earlier, has no valid numbering. It is of interest to find which restrictions need to be placed on a subsystem S' such that if the system S has a valid numbering, then the subsystem S' will also have a valid numbering and vice-versa. If it is possible to reduce the search for a valid numbering to a search across a variety of specific subsystems, it may be simpler to determine if a given system has the MULTISTABLITY property.

Lemma 4.2.3. *Let $S = (N, A, R)$ be an A-R system with a valid A-R numbering F and $F(n) = 0$ for some $n \in N$, then F also a valid A-R numbering for the subsystem $S' = \{N \setminus n\}$*

Proof. This is obvious from the fact that $F(n) = 0$ adds at most a zero to the a and r values for any other node, so its removal from the system will have no effect on the a and r values for any nodes, and thus the numbering remains valid. ☐

Lemma 4.2.3 provides a simple method to determine subsystems with valid numberings given a numbering for the entire system. However, this provides only one numbering for the subsystem and does not give any insight into whether the subsystem could have more than one numbering as any given node may have $F(n) = 0$ for one numbering and $F(n) \neq 0$ for another. Regardless, the proof provides an insight into how changing one node's value affects another node's value and provides some information about potential fixed states.

Definition If $S = (N, A, R)$ is an A-R system. A node $m \in N$ is said to be dependent on $n \in N$ if there exists a path in $A \cup R$ from n to m. If there also exists a path from m to n and $m \neq n$, the nodes are said to be codependent.



(a) A Tree (b) Switch Dependency

Figure 4.3: Dependency Examples

All nodes are dependent on themselves, as the trivial path always connects n to itself. As well, all nodes within a cycle are codependent on each other. However, it is not necessary for two nodes to lie in the same cycle to be codependent, this can be seen by taking the union of two cycles with a single connecting node. All nodes are codependent without being in the same cycle. However, for there to exist codependent nodes, the codependent nodes must lie in a directed cycle. This can be seem from the need for an incoming and outgoing edge for the codependent node.

In figure 4.3a, all nodes are dependent on a central node, and no nodes are codependent as the system is simply a tree radiating from a common node, so there are no directed cycles. In figure 4.3b, all nodes within the cycle on the left are codependent, and are dependent on the toggle switch to the right. This example clearly illustrates why this feature is called dependent. As discussed before, a toggle switch such as that on the right has two possible numberings, and the cycle on the left has no valid numbering, however if the toggle switch is in the state such that the marked node is high, the system has one valid numbering, which is given. Thus, the dynamics of the cycle on the left are entirely dependent on the state of the switch on the right. This leads to a very natural lemma regarding dependency and numberings.

Lemma 4.2.4. *Given an A-R system S and some numbering, F, if the node m's value, $F(m)$ is affected by the changing of another node's value $F(n)$, then m must be dependent on n.*

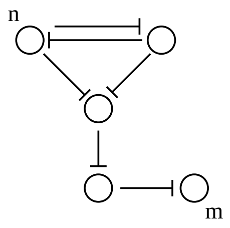


Figure 4.4: Counterexample where m depends on n but is not affected by a change in n

Proof. Let $F(m) = \frac{2a(m)}{a(m)+r(m)}$, where $F(m)$ is affected by the changing of another node $F(n)$.

If $n \to m \in R \cup A$, then there exists a path from n to m, and thus m is dependent on n.

If $\nexists n \to m \in R \cup A$, then for some $m' \in N \setminus m$, $F(m')$ is affected by the adjustment of $F(n)$ where $m' \to m \in R \cup A$. If $n \to m' \in R \cup A$, then there is a path from n to m, and again m is dependent on n. Repeating this argument recursively it is possible to build a path from n to m, and thus show that if $F(m)$ is affected by a change to $F(n)$, then m is dependent on n. □

Corollary 4.2.5. *If a node m is not dependent on a node n in S, an A-R system, then for any numbering F, the value of $F(m)$ is not affected by any change to the value of $F(k)$ for any node k which is dependent on n.*

Proof. This follows naturally from the lemma. □

The converse of this corollary is not true. A simple counterexample is the system such that the only node between m and n is k such that $F(k) = 0$ for all possible numberings, then the value of $F(m)$ can never be affected by any change to $F(n)$. This is shown in Figure 4.4.

Definition A given subsystem $S' \subset S$ is closed, if all nodes are co-dependent in S'. If no nodes are co-dependent, the system is said to be open.

Proposition 4.2.6. *If S is an open A-R system, then there exists exactly one numbering and it can be efficiently calculated.*

Proof. For the case where $S = \emptyset$, this is obviously true.

We may assume that $S \neq \emptyset$. If no nodes are co-dependent, there must exist a set L of nodes which are not dependent on any other nodes. Since these nodes are not dependent on any other nodes, they must have their a, r functions equal to 0, therefore $F(l) = 1, \forall l \in L$. From this, it is simple to calculate the values of the other nodes from this starting point by simply calculating the value $F(m) = \frac{2a(m)}{r(m)+a(m)}, m \in M = \{m \in N \setminus L | \forall n \to m \in A \cup R, n \in L\}$, setting $L = L \cup M$, and repeating until $L = N$.

The accuracy of this algorithm is guaranteed as the numbering is exactly as defined in the definition of an A-R numbering. The efficiency of this algorithm is $O(|N|)$, as it calculates F once and only once for each node. □

First, this states that the MULTISTABLITY problem is trivially false for any genetic regulatory network whose network representation is a tree. It shows that without a closed subsystem within S, the A-R numbering problem is trivial. However, once there is a closed subsystem, and thus a cycle, other possibilities arise. There may exist more than one fixed state, or there may not even be a fixed state for a given A-R system. So, it is necessary to study closed systems to understand the MULTISTABILITY problem. This line of research will be returned to in the Basal Rate-Repression System section of this chapter.

Definition A given subsystem $S' \subset S$ is isolated if no nodes in S' are dependent on nodes in $N \setminus S$.

With this property of subsystems it's possible to relate a valid numbering from S to an isolated subsystem S'. Note that the trivial subsystem $S' = S$ is by definition an isolated subsystem of S.

Proposition 4.2.7. *S has a valid numbering $F : S \to \mathbb{Q}^+$ if and only if all isolated subsystems have a valid numbering.*

Proof. ($\Rightarrow$) Given a valid numbering F for S, and an isolated subsystem S'. By definition S' has no nodes which are dependent on nodes outside of S'. Therefore, it was shown before that if a node m is not dependent on another node n, then $F(m)$ is not affected by any nodes k which are dependent on n. As such if n and all nodes k which are dependent on n are removed, the numbering remains valid for the remaining nodes. Hence, if all nodes $N \setminus S'$ are removed, the numbering remains valid for S'. Thus, if S has a valid numbering, any isolated subsystem S' has a valid numbering.

($\Leftarrow$) Since S is itself an isolated subsystem, if all isolated subsystems have a valid numbering, then S itself must have a valid numbering. $\square$

Definition A given subsystem $S \in N$ is said to be controlled by another subsystem $L \in N$ if there does not exist any path in $R \cup A$ from S to L and there exists a path in $R \cup A$ from L to S.

Definition Given an A-R system, a subset of nodes $S \subset N$ is called a switch if it is isolated, closed and there exists more than one valid A-R numbering for the associated A-R system where N is restricted to S.

An example of a subsystem controlled by a switch in Figure 4.3b with the toggle switch controlling the repressilator. From a biological standpoint being able to find examples of these systems within real-world genetic regulatory networks is valuable from the perspective of finding drug candidates and differentiation factors. If a switch is found that controls a part of the network which directs the behaviour of the cell, it could lead to a variety of drugs and procedures for medical treatments. For example, if a switch was found that turns off the replication genetic circuitry in a cell, it could be used to stop cancerous stem cells from reproducing by forcing them to differentiate into a non-lethal cell type, or simply cause apoptosis (cell death).

These are two very intuitive and helpful definitions to show how subsystems behave and interact. For example, if the PU.1, GATA-1 switch were to be further connected into a larger network, there would be various major subsystems which are directly controlled by the current state of the switch. Knowing the purpose of the subsystems through other techniques would provide a hint as to whether the switch is functionally useful to target with drug treatments. As well, using this idea of subsystems controlled by switches, a large amount of cellular phenomena becomes intuitive. For example, if a known switch is part of a larger switching mechanism, the evolutionary purpose of such a switch would be clear. For example, this method of switches controlling each other is hypothesized to be a mechanism for how cells ensure one-way differentiation down a tree from a single progenitor cell to a variety of stable final states [38].

From this, it becomes obvious that any heuristic approaches to the MULTISTABILITY problem which provide solutions to real world systems will be very important for future medical biological use. This numbering idea provides a method to search for natural switches within the biological networks. Isolated systems are easily detected by using network flow theory and finding one-way cuts within the large flow graph, and then if it is possible to determine the MULTISTABILITY, these isolated systems can be tested to determine if they are switches, and those switches naturally become areas of the genetic regulatory network to concentrate further medical research on.

As well, the behaviour of these A-R systems and A-R numberings are interesting from a purely mathematical perspective. If the conjecture, 4.2.2, provided in this section is indeed true, it will be an exciting result with respect to methods of determining fixed states of these types of ODEs. It provides an inherently mathematical link between a large subset of ODEs and combinatorial theory. Similarly, any way of defining system-based constants for these types of systems would be interesting as they would naturally relate to the Mean Field Model system of ODEs and perhaps even provide insights to the entire field of dynamical systems.

Thus, the A-R numbering provides two exciting possibilities: biologically it provides a simpler system to detect switches and drug targets within the genetic regulatory network, and mathematically it provides an entirely new realm within combinatorics which has connections to dynamical systems theory and network flow theory. In the following section, these A-R systems will be restricted even more so to prove the complexity of the MULTISTABILITY problem and provide some heuristics for detecting switches and MULTISTABILITY within a given genetic regulatory network.

4.3 Steady State Classifications of A-R Systems

Classifications of A-R Systems Based on their numberings, it is possible to classify A-R systems into six types which describe their common dynamics and fixed states from the structure.

COMPLETELY UNSTABLE A system is COMPLETELY UNSTABLE if there exist a switch $S \in N$, and for any valid value of the switch, the entire system S has no valid A-R numbering.

Figure 4.5 shows an example of a system which is completely unstable. It has a toggle switch connected to two different repressilators. Compared to a real world system, this would be similar to having a switch connected to two engines, as the switch changes position, a different engine starts. Dynamically, when the switch is in one position, one repressilator is oscillating, and vice versa. Using the PU.1, GATA-1 example, since the state of the PU.1, GATA-1 switch fundamentally determines the long-term dynamics of the system, it is likely that in the larger genetic regulatory network, there exists two "engines", which are either active or inactive based on the PU.1, GATA-1 state. However, at this time, this is simply hypothesized.

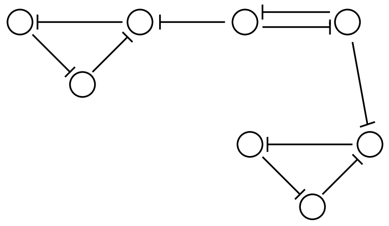


Figure 4.5: Example of COMPLETELY UNSTABLE

COMPLETELY STABLE A system is COMPLETELY STABLE if there exists a switch $S \in N$, and for any valid value of the switch, the entire system S has at least one valid A-R numbering.

Figure 4.6 demonstrates a system which is completely stable. It has a switch connected to the repressilator such that regardless of the state of the switch, the repressillator remains inactive. In many circumstances, like the one in the figure, this increases the size of the switch without having any real effect on the overall dynamics of the system. However, a tree of switches can also be completely stable, and thus this would have the effect of pointing out cascading switch systems. Similar to how once PU.1 and GATA-1 become unstable and choose between themselves, they are believed to make other self-activating switches down the line unstable so they need to choose between two or more possible final states, thus leading to the one-way differentiation of cell types.

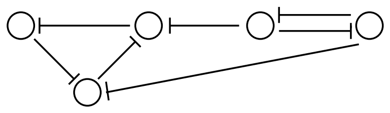


Figure 4.6: Example of COMPLETELY STABLE

VARIABLY UNSTABLE A system is VARIABLY UNSTABLE if there exists a switch $S \in N$, and for at least one valid value of the switch, the entire system S will have no valid A-R numbering.

VARIABLY STABLE A system is VARIABLY STABLE if there exists a switch $S \in N$, and for at least one valid value of the switch, the entire system S has a valid A-R numbering.

These two classes are two sides of the same coin. While they are not equivalent to each other, many different types of systems lie in both of these classes, see Figure 4.7 for an example of this. These types of systems have similar implications to the completely unstable and completely stable classes. Except, the state of the switch can, in certain circumstances, determine if a system within a genetic network is active or inactive at any time. Outside of the context of a genetic regulatory network, one example of a system that lies in both of these regimes would be similar to a car's ignition and engine. When the key is in the off position, the system is completely stable and nothing is happening:

when the system is turned to the on position it becomes unstable and the engine starts to work.

In a genetic regulatory network context, if there was a switch which controlled whether or not a cell would periodically reproduce itself, it likely would be such that in one state it repressed an unstable subsystem to force it to stablize, and in the other state would release that subsystem to become active and unstable and thus periodically perform some action. In a way, these are the most interesting of the subsystems within the genetic regulatory network as these systems connect a switch's state with its function. If it were possible to find these rapidly, it would be possible to look at the genes directly affected by the switch and determine their functionality in the larger system.

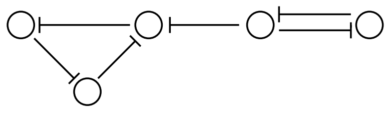


Figure 4.7: Example of VARIABLY UNSTABLE / VARIABLY STABLE

UNSTABLE A system is UNSTABLE if there does not exist a switch $S \in N$, and the entire system S has no valid A-R numbering.

These systems are remain active with no stable fixed states until the cell dies or something changes the chemical reaction rates within this system. In many cases, these behave like a genetic motor doing some form of periodic behaviour.

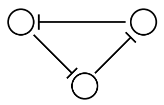


Figure 4.8: Example of UNSTABLE

STABLE A system is STABLE if there does not exist a switch $S \in N$ and there exists a valid A-R numbering.

A stable system does not have much effect on the entireity of the network. If it is determined that a system is stable, it means that it will, in general, always return to a singular state and, without noise, not have any notable dynamics.

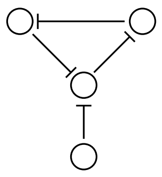


Figure 4.9: Example of STABLE

This provides a natural division of this class of systems which relates their dynamical behaviour to their network structure. A given A-R system may be part of more than one of these sets, as shown in the following lemma.

Lemma 4.3.1. *The following are true:*

1. *COMPLETELY STABLE* $\subset$ *VARIABLY STABLE*
2. *COMPLETELY UNSTABLE* $\subset$ *VARIABLY UNSTABLE*
3. *VARIABLY STABLE* $\cap$ *VARIABLY UNSTABLE* $\neq \emptyset$

Proof. Parts 1 and 2 are self evident, since if all possible switch states have the STABLE (UNSTABLE) property, then there must exist at least one which has that property.

Part 3 is demonstrated by figure 4.7 $\square$

In the general case, determining whether a given A-R system lies within any of the above sets is equivalent to finding an algorithm to solve NP-Complete problems. This will be shown in the following section by looking at a subset of A-R systems which have more intuitive fixed state properties.

4.4 Basal Rate-Repression Systems

It is helpful to simplify the MULTISTABILITY problem further to understand it better. The next simplification is the removal of the activation side of the A-R System entirely. This is equivalent to a genetic regulatory network where all nodes have a high basal rate. In the ODEs, this translates into a high η and all observable dynamics are related totally to repression. This is not an uncommon way to examine genetic regulatory dynamics. Many of the seminal papers over the last decade examined systems in which only repression was the primary factor in any dynamics. The toggle switch and repressilator are good examples of this.

Definition A basal-repression system (B-R system) is an A-R system (N, A, R) for which $A = \emptyset$.

Lemma 4.4.1. *The only possible values for a valid A-R numbering F for a B-R system have $F : N \to \{0, 1\}$.*

Proof. For any node n in a B-R system with a given valid numbering F, there are two possibilities. Either $\forall m \to n \in R, F(m) = 0$, in which case $r(n) = 0, a(n) = 0$, thus $F(n) = 1$, or $\exists m \to n \in R, F(m) > 0$, thus $r(n) > 0, a(n) = 0$, and thus $F(n) = 0$.

Thus, there are only two possible values for any valid A-R numbering of a B-R system, 0 and 1. □

This naturally leads to the following important proposition.

Proposition 4.4.2. *Given a B-R system $S = (N, R)$, there exists an A-R numbering F for such a system if and only if the following conditions are satisfied:*

1. *$\forall n$, if $\exists m \to n \in R$ where $F(m) = 1$ then $F(n) = 0$*
2. *otherwise $F(n) = 1$*

Proof. Well, as the previous lemma demonstrated, for any valid numbering F for a given B-R System $S = (N, R)$ has the feature that $F(n) \in \{0, 1\}$. Hence, it is necessary to look at two possibilities for any $F(n)$ given a valid A-R numbering (note that $a(n) = 0, \forall n \in N$):

1. $\forall m \to n \in R, F(m) = 0 \Leftrightarrow F(n) = 1$, since $r(n) = 0$ and thus $F(n) = 1$, since F is a valid A-R numbering.
2. $\exists m \to n \in R, F(m) = 1 \Leftrightarrow F(n) = 0$, since $r(n) > 0$ and thus $F(n) = 0$, since F is a valid A-R numbering.

Thus, if F is a valid numbering, the above axioms are obeyed for F. □

The previous conjecture, 4.2.2, stated that all stable fixed states were contained within the set of all associated possible numbering. Assuming that conjecture is correct, another conjecture arises for B-R systems. The counterexamples for the A-R system where it would provide a numbering that was associated to an unstable fixed state given certain rate parameters depending on certain conditions where mutual repression is cancelled by mutual self-activation. These types of circumstances simply do not happen for B-R systems as the restriction fundamentally binarizes the numberings. Thus, it becomes reasonable to state a stronger conjecture for B-R Systems.

Conjecture 4.4.3. *All stable fixed states for the MFM for B-R systems have an associated possible numbering for the graph and vice versa.*

Lemma 4.4.4. *Given a B-R system where R is only a path of length k (See figure 4.10). The only valid numbering is $F(m) = m(mod\ 2)$, where m is the m^{th} node in the path.*

Proof. $k = 0$ is obvious. $k > 0$ will be proved by induction on k.

For $k = 1$, there is only one node in the network $n = 1$, by definition this node has no other nodes pointing at it, so the first axiom of a B-R numbering states that $F(n) = 1 = 1(\text{mod } 2)$. Thus, this is true for $k = 1$.

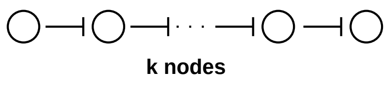


Figure 4.10: A repression path of length k

Assume true for $k = l$. For $k = l + 1$, no other nodes are dependent on the node $(l+1) \in N$, therefore any valid numbering, F, must be valid for $S \setminus (l+1)$, and the only valid numbering is $F(m) = m(\text{mod } 2)$. Therefore $F(l) = l(\text{mod } 2)$. Therefore $F(l+1) = l + 1(\text{mod } 2)$, since the only node with a directed edge at $(l+1)$ is l. $\square$

This proof leads to an interesting and simple result showing the power of this mathematical object for detecting fixed states. It is already known in the literature that a self-repression cycle will have more than one stable fixed state if the cycle length is even, and will have no stable fixed states if the cycle length is odd. However, unlike the other proofs, this result trivially falls out without any complicated dynamical systems analysis. This is because any isolated path in a system S must have an alternating numbering of $1, 0, 1, 0, \ldots$.

Proposition 4.4.5. *Suppose we are given a B-R system which is a cycle of length $k > 1$ (see figure 4.11). If k is odd, the system has no valid numbering. If k is even, the system has only two valid numberings.*

Proof. First note that if $k > 1$ and there exists a valid numbering F, there must exist a node n such that $F(n) = 1$, and another node m, $n \to m$ such that $F(m) = 0$.

Take the case where S is a cycle of length k, where k is odd. Assume for the sake of contradiction that a valid numbering F exists. Thus, for some node n, $F(n) = 0$. This is because for some $m \to n \in R$, $F(m) = 1$. Therefore, from lemma 4.2.3, F will be a valid numbering for $S \setminus \{n\}$, a path of length $k - 1$. Relabel the nodes in the path such that the first node in the path is 1, and the final node in the path is $k - 1$. Since k is odd,

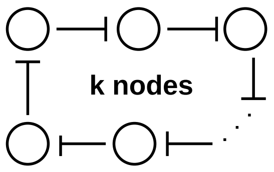


Figure 4.11: A repression cycle of length k

$k-1 = 0 (\text{mod } 2)$. However, $F(m) = 1$. Therefore there is a contradiction and F cannot be a valid numbering for S. Therefore, if k is odd, the system has no valid numbering.

Now, take the case where S is a cycle of length k, where k is even. It is possible to create a numbering F for S, where each node alternates between $0, 1$. Therefore, one of the nodes, n, must have $F(n) = 1$, and thus the node before it in the cycle, m, must have $F(m) = 0$. This numbering exists if and only if $S \setminus \{m\}$ has the same numbering. Well, the node l with $l \to m \in S$ has the value $F(l) = k-1 (\text{mod } 2) = 1$. Therefore $F(m) = 0$. Thus, this is a valid numbering for S. Similarly, it is possible to create a second numbering G, where $G(n) = F(n) \oplus 1$, and hence $G(n) = 0$. Thus S has two valid numberings. □

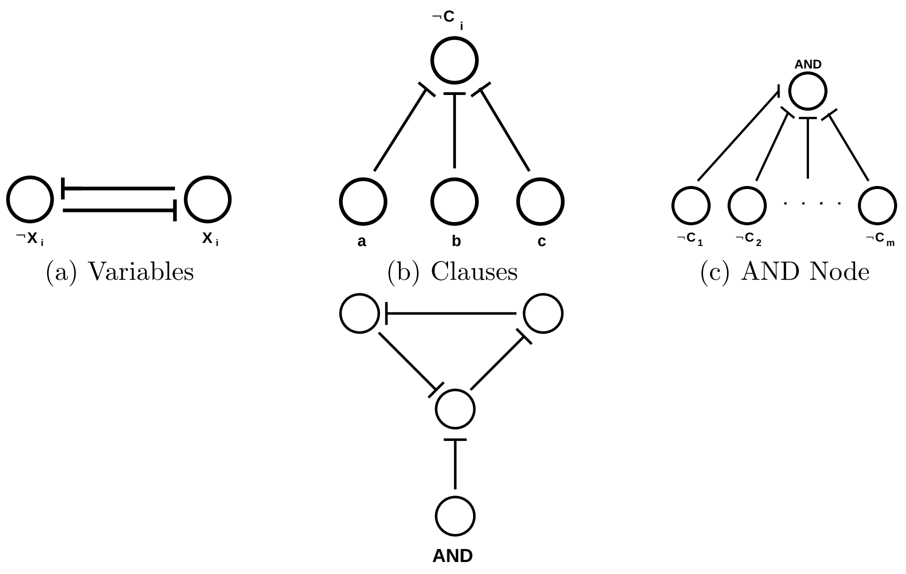
Lemma 4.4.6. *Given a B-R System with a node n and edges $m_i \to n \in R, 1 \leq i \leq k$, for any valid numbering F,*

$$F(n) = \bigwedge_{1 \leq i \leq k} \neg F(m_i)$$

.

Proof. If $\exists m_i, f(m_i) = 1$, then $\bigwedge_{1 \leq i \leq k} \neg F(m_i) = 0 = F(n)$, by proposition 4.4.2. Similarly if $\forall m_i, F(m_i) = 0$, then $\bigwedge_{1 \leq i \leq k} \neg F(m_i) = 1 = F(n)$, again, by proposition 4.4.2. Hence, this lemma is true. □

Now, we come to the central theorem of this thesis where we demonstrate that the MULTISTABILITY problem is NP-Complete, and thus fundamentally a difficult problem



(a) Variables (b) Clauses (c) AND Node

(d) AND Node

Figure 4.12: Elements of 3-SAT reduction

to solve in the most general case.

Theorem 4.4.7. *Determining if a B-R system is VARIABLY STABLE (UNSTABLE) is NP-Complete.*

Proof. Given an instance of 3-SAT problem $\{x_i\}_{1\le i\le n}, \bigwedge_{1\le i\le m} C_i$, where C_i are the independent clauses of the satifiability problem, create an B-R system S as follows.

For each variable x_i, create 2 nodes in the B-R system $x_i, \neg x_i$ which are mutually repressing as shown in figure 4.12(a). For all clauses $C_i = (a \vee b \vee c)$, where a, b, c are the values for the clause (e.g. for $C_l = (x_4, x_2, \neg x_1)$, $a = x_4, b = x_2, c = \neg x_1$), create a clause node "$\neg C_i$" with edges in R coming from the nodes a, b, c. This is shown graphically in figure 4.12(b).

This system currently has $2n + m$ nodes and $2n + 3m$ edges. Add a node "AND" to the system and add an edge from all of the clause nodes $\neg C_i$ to this node (Figure 4.12(c)). Finally, add a repressilator to the system and connect "AND" to one node of the repressilator (Figure 4.12(d)).

The final count for the size of this system is now $2n + m + 4$ nodes and $2n + 4m + 4$

edges, a linear increase in the size of the 3-SAT problem. Therefore we have a polynomial reduction. Now, this proof will prove that being able to state whether this system is VARIABLY STABLE is equivalent to showing the 3-SAT instance is satisfiable. This comes from the fact that the repressilator will remain unstable unless the $F(AND) = 1$. So it is necessary to show that the value of $F(AND) = \bigwedge_{1 \leq i \leq m} F(C_i)$. From the previous lemma

$$F(\neg C_i) = \neg F(a) \wedge \neg F(b) \wedge \neg F(c) = \neg(F(a) \vee F(b) \vee F(c)) = \neg C_i$$

Similarly, from the lemma it is possible to see for $F(AND)$.

$$F(AND) = \bigwedge_{1 \leq i \leq m} \neg F(\neg C_i) = \bigwedge_{1 \leq i \leq m} \neg\neg C_i = \bigwedge_{1 \leq i \leq m} C_i$$

Thus, there is a numbering with $F(AND) = 1$ if and only if the 3-SAT problem given is satisfiable. Therefore, the repressilator remains unstable for all possible values of the switches unless the 3-SAT problem is satisfiable. Thus, the problem, "Is a given B-R system (N,R) VARIABLY STABLE?" is NP-Complete. □

Since the B-R systems are a subset of the A-R systems, this implies that the problem "Is a given A-R system (N,A,R) VARIABLY STABLE?" is also, at minimum, NP-Complete. The similar problem "Is a given A-R system VARIABLY UNSTABLE?" is also NP-Complete. As can be seen by adding an extra node $\neg AND$, adding an edge from AND to $\neg AND$, connecting $\neg AND$ to the represillator and removing the edge from AND to the represillator. Thus if $F(\neg AND) = \neg F(AND) = 0$ the system is UNSTABLE for a given settings for the switches.

For an A-R system to be guaranteed to be NP-Complete, this must be limited to finite possible numberings. Otherwise, the non-deterministic nature of the problem will not be bounded. If there was an A-R system with unbounded possible numberings, then the problem would lie within the capability of unbounded non-deterministic Turing machines. A set, which was discussed before, is capable of handling the halting problem. In fact, since the problem is NP-Complete for the bounded case, it is likely that it is uncomputable in the general case for unbounded numberings.

Regardless, the problem remains not efficiently solvable in the general case.

Corollary 4.4.8. *Stating a given B-R system is COMPLETELY STABLE or COMPLETELY UNSTABLE is CoNP-Complete*

Proof. This follows naturally from theorem 4.4.7, as COMPLETELY STABLE = $\neg$ VARIABLY UNSTABLE and COMPLETELY UNSTABLE = $\neg$ VARIABLY STABLE. $\square$

So, what do these two proofs imply? First they imply that these systems are highly complex entities, even when restricted to only repression, simply because the entire complexity class NP is contained within their fixed state dynamics. Sadly, this also shows that the problem of MULTISTABILITY is a fundamentally difficult problem, assuming $NP \neq P$.

Corollary 4.4.9. *The MULTISTABILITY problem is at least NP-Complete.*

Proof. Given an instance of a 3-SAT problem, $(\{x_i\}_{1 \leq i \leq n}, \{C_i\}_{1 \leq i \leq m}$ it is possible to convert it to a 3-SAT problem where the number of possible satifiable states are ≥ 2, if and only if the original 3-SAT problem is satisfiable. This is done by adding one new variable y and creating one new clause $C_{m+1} = (y \vee y \vee y)$. Obviously this system doubles the number of satisfiable Boolean states for the original 3-SAT problem, and remains NP-Complete without increasing the size of the instance by more than a constant amount.

With the reduction from the theorem 4.4.7, it is possible to create an A-R system $S = (N, A, R)$, for which there is more than one stable state if and only if the original 3-SAT problem is satisfiable. Thus, if it is possible to state the existence of multiple stable fixed states for the associated Mean Field Model to the A-R system, then it will provide an answer to the associated 3-SAT problem.

Assuming that the conjecture that the A-R numbering gives all stable fixed states is true, the problem of determining whether a given system is MULTISTABLE is guaranteed to be NP-Complete, and likely uncomputable for systems where there are an unbounded number of possible stable states. $\square$

This corollary shows that the MULTISTABILITY problem is provably difficult and likely impossible. Finding an algorithm for the most general case will at very least solve any problem in the NP complexity class. Sadly, it also points out that even restricting the system in such a way that the stable states are finitely enumerable the problem remains difficult as long as repression loops are maintained. Thus, it is necessary to look at possible heuristics and subsets of the B-R systems (and the A-R systems) for which it is possible to get a handle on the MULTISTABILITY problem and see if there is a point where the system admits mathematical examination.

Chapter 5

Restrictions and Heuristics

This chapter discusses some further restrictions to the B-R system which allow for further understanding of the MULTISTABILITY problem. As well, these restrictions give special cases for which class membership is easily determined. Unfortunately, very few systems exist after the restriction makes the system efficiently solvable. In fact, even with the restriction to closed B-R systems, the problem remains NP-Complete. However, regardless of the NP-Complete nature of the system, it is useful to look at restrictions to learn more about where the complexity arises, and attempt to find some method to handle a large class of systems more easily.

From these efficiently computable cases, a rough heuristic algorithm is developed which easily determines membership for a variety of cases and becomes inefficient based on how many disconnected even cycle only subsystems exist within the full system., thus providing not only an angle of attack on the problem, but a way to bound the problem given a starting set of disconnected even cycle only subsystems.

5.1 Closed B-R Systems

Closed B-R systems have been defined as systems where there exists a path from any node to any other node. This implies that every node lies in a cycle and the system is composed of a series of overlapping cycles. In the case where all of the cycles have the same parity (either even or odd), the question of membership into the various classes of systems is easily solved. In the case where one odd cycle is connected to a system of even cycles, the question of membership remains easy. In fact, if only odd cycles are added to the system of even cycles, the problem remains simple. What this provides is an insight

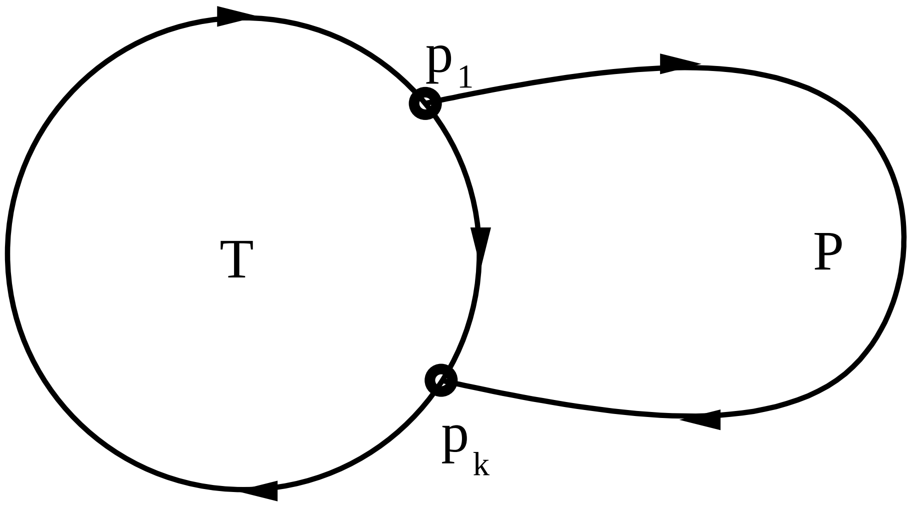


Figure 5.1: A path connecting to the system T, creating a new cycle

as to where the complexity for the MULTISTABILITY problem lies.

Note that once there are $n >> 1$ systems of even cycles within a larger system, the problem becomes intractable.

Proposition 5.1.1. *If S is a closed B-R system where all cycles are even then the system is COMPLETELY STABLE with exactly two different numberings.*

Proof. Given a system S where all cycles are even, it is possible to construct two different numberings which are valid for the system.

Take any cycle $C \subset S$ and create a new system $T = C$. T will have two valid numberings since it is an even cycle as per a previous lemma.

Repeat the following actions recursively until all nodes have been added to the system T.

Take any other path of length k, which connects from one node of this system to another node as shown in Figure 5.1, and continue this numbering through that path. It is claimed that there remains two valid numberings for this new system.

Now, there are two possibilities: one is that the cycles will connect in such a fashion that the path connects with an alternating numbering which does not interfere, in which case there will still be two valid numbering by taking the inversion of the current numbering (i.e., 0 goes to 1, and 1 goes to 0.) Thus the recursive assumption that T has two numberings remains valid. This possibility includes the situation where the path starts and ends at the same node. Since the path would then have to be of even length, it will have an alternating numbering which does not conflict.

The other is that the path will start at $n = 0$ or 1, be of length k, and reconnect to the existing system at $m = n + k \bmod 2$. In this case, there will only be one valid numbering for the system, and so the system cannot be COMPLETELY STABLE. However, this case implies the existence of an odd cycle. For the path to connect in this fashion, there must be a different parity between the length of the path P and the distance of the path L between the two nodes within the original system T that connect the path to the graph. The two nodes which connect the path to the graph are part of some cycle C of even length, by assumption. So, if a new cycle is created by taking $C \setminus L \uplus P$, this new cycle must have odd length. So therefore this is absurd given the assumptions that all cycles are even.

Therefore, if the system is closed and all cycles are even, then the system is COMPLETELY STABLE with two numberings. □

Proposition 5.1.2. *Given a closed B-R system such that all cycles are odd, the system is UNSTABLE*

Proof. This will be done through proof by contradiction. Let S be a closed system consisting entirely of odd cycles. First, assume that there is a valid numbering for S.

For a numbering to exist, there would have to be two consecutive zeroes at the starting point of connection for each cycle in the numbering, with the second zero being the node of connection. This is because for any numbering, if an odd cycle does not have at least two zeros in a row, then it must have two ones in a row somewhere, which is impossible.

To establish the contradiction, take each cycle C and examine a double zero within the numbering. If this cycle is connected by a single node n to all other cycles in the rest of the graph, the contradiction can be found by doing the following with the numbering on the subsystem $S' \setminus C_1 \cup \{n\}$, where n is located at the double zero. So the numbering must remain valid on this subsystem, because of the zero at the connecting point.

However, if the double zero occurs due to a connection to another cycle C_2 with a path P shared between them, having starting node p_1 and ending node p_k, $p_1 \neq p_k$, then the double zero must occur at the first node of that path. Since the first node of the path is zero, then any numbering on the system S must remain valid on the subsystem $S' = \{S \setminus P\} \cup \{p_1, p_k\}$.

Note that these subsystems all have at least one cycle remaining. Taking the subsystem, for which the numbering remains valid, repeat this procedure until there is a final subsystem with only one cycle. Now, this cycle must be even since it must have a valid numbering and odd cycles have no valid numberings. Thus, there exists an even cycle in the original system contradicting the assumption that all cycles are odd.

Therefore, a system consisting of only odd cycles must be UNSTABLE. $\square$

Proposition 5.1.3. *Given a closed B-R system of only even cycles, if a path is added which creates an odd cycle, then the system is either STABLE or COMPLETELY STABLE with two numberings.*

Proof. Well, let T be the closed system consisting of only even cycles. Therefore T has two possible numberings from the previous proposition. Add a path to the system which creates an odd cycle. Let p_1 be the start node of this path and p_k be the end node of this path. Now, as was seen in the constructive proof that a system with all even cycles has two numbers where every node has value 0 for one numbering and value 1 for the other. So by choosing the numbering where p_k has value 0, then no matter what numbering the path has, given p_1's number, it cannot conflict at p_k, and thus will be a valid numbering. Therefore, the system is at least STABLE, and perhaps COMPLETELY STABLE.

It is easy to test if the system is COMPLETELY STABLE by flipping the value of p_1 and verifying by following the numbering through the path and the system, and checking if p_k remains 0 and if any node pointing at p_1 has a conflict by having two ones in a row. If there is no conflict, and p_k remains 0, then the system is COMPLETELY STABLE with two numberings where p_k is 0 in both numberings, otherwise the system is simply STABLE. □

The previous proof seems to offer a method to efficiently check if a system is either STABLE or COMPLETELY STABLE by adding paths progressively to some initial cycle. However, even though it seems that it should be possible that closed B-R systems are simpler to handle, the problem of determining their dynamic attributes from their structure still remains NP-Complete.

Theorem 5.1.4. *Given a closed B-R system, solving whether the system is STABLE or COMPLETELY STABLE is NP-Complete and UNSTABLE is coNP-Complete.*

Sketch. The full details of this proof are omitted simply because this is a natural extension of the previous completeness proof where repression edges are added from the final repressilator to all of the variables so that if the repressilator is not active, the new edges will have no effect on the variable. This makes the system closed, while still making membership in any of the classes dependent on the existence of a solution to the related 3-SAT problem. □

In spite of this theorem, the proposition regarding even/odd cycles can still be used to develop a heuristic algorithm to test whether a closed system lies within STABLE, COMPLETELY STABLE, or UNSTABLE, and thus whether it is MULTISTABLE or not. This is valuable, even though it can take a long time to run for certain cases.

5.2 A Heuristic Approach

Even though the final theorem of the previous section implies that, even with this level of restriction, membership problems remain intractable, there can be some hope. Using the even-cycle system propositions from before, it is possible to develop a heuristic algorithm which will work efficiently in cases where there are few disconnected even cycle subsystems, and only becomes exponential based on the number of disconnected even-cycle subsystems within the system.

The heuristic algorithm works as follows: Given a closed B-R system S, do the following:

1. Find a set of disconnected closed even-cycled only subsystems contained within S, (ie. $E = \{S_1, S_2, \ldots, S_n\}, S_i \subset S, S_i \cap S_j = \emptyset, \forall S_i, S_j \in E$, S_i is closed, and for any cycle $C \in S_i$, $|C|$ is even. Note, the choice of this set will strongly influence the runtime of the algorithm and thus there is room for speed and memory usage improvement here.

2. It is known for each of these even-cycled systems that there are two possible numberings. Since they are seperated and thus do not influence eachother, there are currently 2^n possible numberings for the system consisting only of E.

3. For each of the subsystems S_i, create a list containing the two numberings N_i and do the following.

   (a) If there exists a path P which starts and ends in S_i, and has no nodes lying in any other $S_j \in E$.

   (b) Add P to S_i, and check to see which valid numberings have p_k, the last node of P, set to 0. These all remain valid numberings, and need not be changed beyond adding P to them and setting the path's node values as appropriate. This is similar to the all evens + one odd proposition from earlier.

(c) If p_1 in the path is 0 for all possible known numberings, test with each numbering to see if setting its value to 1 also leads to a valid numbering. If so, add those numberings to the list of valid numberings for S_i.

(d) Test each numbering by setting p_k to 1 and changing other node's values as appropriate and seeing if this implies p_k must also be 0. If it doesn't, then this is also a valid numbering

(e) Finally, if there are no valid numberings with p_k being 0, then test if setting p_k to 0 leads to a valid numbering.

(f) Repeat this until there are no P which start and end in S_i with no nodes in other S_j.

4. At this point each S_i will have some set of numberings N_i, possibly an empty set. Now, take two subsystems, S_i, S_{i+1}, and repeat the following by taking the cross product of their sets of valid numberings $N_{i,i+1} = N_i \otimes N_{i+1}$ and create a set of systems $S_{i,i+1} = S_i \cup S_{i+1}$

5. Repeat the previous two steps until there is only one system left, the original system. Now, you will have a complete list of possible numberings for the original closed system.

At this point, it is possible to determine which dynamic category the original system S lies in. As well, with this tool, it becomes possible to determine these properties for any non-closed systems as all that is needed is to run this algorithm for each subsystem which acts as a controller, and then use the list of possible numberings for that controller to pre-set full system values and repeat for controlled elements. Obviously, as there are more controllers, the number of runs grows rapidly. This is, in essence, why the MULTISTABILITY problem remains so difficult for systems which are not necessarily closed. However, it is useful to note that determining if a non-closed system is unstable is as simple as determining if any of the controller subsystems are unstable, since once they are unstable, they can never be stablized.

In a closed system, the runtime is mostly determined by the size of the initial set E. As an exponential combination of numberings will need to be considered as the sets S_i, S_{i+1} are joined in step 4. Interestingly, this also points to the importance of even cycled systems within a large genetic regulatory network since these systems are what create the complexity within stable subsystems.

The PU.1, GATA-1 system is a known even cycled switch within the genetic regulatory network. However, there may exist far larger even cycled switches which, when targeted correctly, can lead the cell to act in incredibly beneficial ways. As discussed in the introduction, it would be reasonable to believe that the methodology to get to the pluripotent cell type essentially does exactly this. By targeting a variety of genes within a large switch, it becomes easier to push it over the breaking point and make the cell revert to an earlier state. From a medical perspective knowing that the even cycled systems are likely to be the important systems to target removes a large swath of targets for drug therapy.

From a biocomplexity perspective, this heuristic and the realization that the even cycles are likely the important cycles with regards to the complexity of behaviour for stability and genetic memory provides a small insight into how complex biological systems work and maintain state. This is a natural continuation to the empirical fact that positive feedback loops provide switch-like behaviour within GRNs.

Chapter 6

Conclusions and Discussion

Any problems regarding genetic regulatory networks will be inherently intractable due to the overwhelming complexity of the chemical systems involved. However, by ignoring a lot of extraneous behaviour and building a combinatorial model based on the Mean Field Model in order to solve a specific problem, the MULTISTABILITY problem, it becomes possible to determine certain interesting systems, map the difficulty of the problem to other known problems, and attempt to create heuristics which solve it in special cases. However, it ended up that even with the simplified model the problem remains intractable in the most general case. Although, the insight provided by the various minor proofs does allow for heuristics and restrictions to be developed which give solutions rapidly for particular cases.

Biologically, this method for gaining some intuition into genetic regulatory networks could be immediately applied to known genetic regulatory network connections to search for possible genetic switches, like the PU.1, GATA-1 switch. Since it is known that even loops have the most likelihood of being multistable, searching for small even loops within the network could be an effective way to find switches. As discussed, this is already being done to a small extent with two gene toggle switches. However, there is no reason that larger, even-cycled, closed subsections could not also be targets for drug candidates. Especially since having more than one gene to target should make it easier to force the entire switch to flip over. The restrictions in this paper, while based on some popular thought in biology, are fairly arbitrary. Restricting to repression-only systems makes the mathematics easier, but ignores the potential for strong activation to significantly change the dynamics. If mathematical research was concentrated into other possible restrictions which better reflect real world networks, it would provide better heuristics for finding multistable subsystems. Similarly, extending the B-R system results to A-R systems to some degree would be valuable.

Mathematically, the methods raised open up a fascinating new way to study certain aspects of mutually interacting dynamical systems. As discussed, the A-R numbering problem could be extended and adjusted for different dynamical systems in several ways. First, weights could be added to the edges and the f-function adjusted appropriately. Second, the $f, a,$ and r-functions within the problem could be modified to compensate for another interacting dynamical system. The particular functions were chosen to be appropriate for the set of interactions seen by the Mean Field Model system of ODEs. It is trivial to adjust them in order to reflect other systems of ODEs which have nodes interdependent in a common fashion.

Finally, the A-R system is only a system of two digraphs. As was seen with B-R systems, fewer digraphs can be easily used to adjust the system. However, it is also possible to add more digraphs in order to deal with more complex interactions within the ODEs. With all of these methods of extending the A-R numbering, it becomes possible to map dynamics with the property of having similar interactions between nodes to the structure of a system of digraphs. A fascinating way to study certain, common dynamical systems.

Even though the model can be extended to cover even more systems, it is also interesting to find out how many systems the A-R or B-R systems as they are currently defined could simulate. If the subset of possible dynamical systems is restricted, it provides an insight on the restricted nature of genetic regulatory network dynamics. On the hand, if the set of possible dynamical systems covers a wide swath, it invites the possibility of developing more techniques with the A-R numbering to find structural properties of dynamic systems. In either case, there is potential for some exciting mathematics.

While most of the evidence seems to imply that conjectures 4.2.2, 4.4.3 are valid, it would probably be an interesting exercise to obtain a definitive answer to this question. Any answer would likely provide an interesting technique for acquiring stable fixed points without directly referring to the commonly used methods already in existence. If either conjecture proves false, the exceptional cases could give some insight about chaotic feedback systems since the counterexample's fixed point may have some unusual properties. Regardless of the truth of these conjectures, the question asked seems to open up a new connection between combinatorics and dynamics.

Due to the relations between the B-R system and bounded non-deterministic problems, and the A-R system, and potentially unbounded non-deterministic problems, there is also a route of investigation for a computational theorist. As restrictions are added to the system, the system becomes progressively easier to compute answers for. Hence, it would be interesting to see if other restrictions could be added which make the problem fit into other interesting complexity classes like bounded probabilistic polynomial. As the initial system is continuous but can be made discrete. It may even provide a transition from continuous computational problems to discrete ones.

As the statistician George E.P. Box stated, "all models are wrong, but some are useful." The Mean Field Model used to develop the A-R system doesn't incorporate every complexity of the associated real world system and the A-R system further simplifies it. However, this does not imply that the tools are not useful as an intuition or will not lead to interesting mathematics on their own. As the intractability of biological systems is made more apparent, having good computational models can be good for simulation, and having simple intuitive models can be even more useful for simple understanding. It is the view of the author that further research to find simpler mathematical structures which still maintain reasonable similarities to the real world system is necessary and this thesis is a small first step towards this end.

As stated in the introduction, sometimes it is better to start from the problem rather than defining the system as whole. By developing a model with the goal of gaining a better handle on the problem, more intuition about the system can be gained, even though the model itself is not a perfect representative. By having a better idea of MULTISTABILITY and other biological problems, better models to study them with, and methods which map them to other known computational problems, it becomes possible to find vectors of attack towards solving deep biological and medical problems. Similarly, by developing these models solely to solve said problems, rather than developing the models to perfectly match reality, new tools and connections will necessarily also be developed. Hence, these techniques not only drive biological knowledge forward, but mathematical insight as well.

Bibliography

[1] S. Aaronson et al. The complexity zoo. World Wide Web electronic publication, 2008. Retrieved July 29, 2008.

[2] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[3] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell, Fourth Edition.* Garland, 2002.

[4] M. Andrecut and S. A. Kauffman. Mean field model of genetic regulatory networks. *New Journal of Physics*, 8:148, 08 2006.

[5] Y. Arinobu, S. Mizuno, Y. Chong, H. Shigematsu, T. Iino, H. Iwasaki, T. Graf, R. Mayfield, S. Chan, P. Kastner, and K. Akashi. Reciprocal activation of gata-1 and pu.1 marks initial specification of hematopoietic stem cells into myeloerythroid and myelolymphoid lineages. *Cell Stem Cell*, 1(4):416–27, October 2007.

[6] P. Bachmann. *Die Analytische Zahlentheorie. Zahlentheorie, pt. 2.* B. G. Teubner, Liepzig, 1894.

[7] T. Baker, J. Gill, and R. Solovay. Relativizations of the $\mathcal{P} =? \mathcal{NP}$ question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[8] D. Bienstock and M. A. Langston. Algorithmic implications of the graph minor theorem. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science, Volume 7, Network Models*, volume 7, chapter 8, pages 481–502. Elsevier, 1995.

[9] R. Blelloch, M. Venere, J. Yen, and M. Ramalho-Santos. Generation of induced

pluripotent stem cells in the absence of drug selection. *Cell Stem Cell*, 1(3):245–7, September 2007.

[10] J. M. Bower and H. Bolouri. *Computational Modeling of Genetic and Biochemical Networks (Computational Molecular Biology)*. The MIT Press, 2004.

[11] R. Bundschuh, F. Hayot, and C. Jayaprakash. The role of dimerization in noise reduction of simple genetic networks. *Journal of Theoretical Biology*, 220(2), January 2003.

[12] M. Carpentieri. On the simulation of quantum turing machines. *Theoretical Computer Science*, 304(1-3):103–128, 2003.

[13] M. Chaves, R. Albert, and E. D. Sontag. Robustness and fragility of boolean models for genetic regulatory networks. *Journal of Theoretical Biology*, 235:431–449, Jan 2005.

[14] J. L. Cherry and F. R. Adler. How to make a biological switch. *Journal of Theoretical Biology*, 203(2):117–133, March 2000.

[15] A. Church. A note on the entscheidungsproblem. *The Journal of Symbolic Logic*, 1(1):40–41, March 1936.

[16] S. Cook. The p versus np problem. In *Clay Mathematical Institute; The Millennium Prize Problem*, 2000.

[17] I. G. Cowell. Repression versus activation in the control of gene transcription. *Trends in Biochemical Science*, 19(1):38–42, January 1994.

[18] W. W. Daniel. *Biostatistics: A Foundation For Analysis In The Health Sciences.* John Wiley & Sons Inc, 8 edition, 4 2005.

[19] H. De Jong, J. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathematical Biology*, 66(2):301–40, March 2004.

[20] B. Derrida and Y. Pomeau. Classical diffusion on a random chain. *Physical Review Letters*, 48(9):627–630, Mar 1982.

[21] V. Devloo, P. Hansen, and M. Labbé. Identification of all steady states in large networks by logical analysis. *Bulletin of Mathematical Biology*, 65(6):1025–1051, November 2003.

[22] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[23] S. Durual, A. Rideau, S. Ruault-Jungblut, D. Cossali, P. Beris, V. Piguet, and T. Matthes. Lentiviral pu.1 overexpression restores differentiation in myeloid leukemic blasts. *Leukemia*, 21(5):1050–9, May 2007.

[24] J. Edmonds. Paths, trees, and flowers. *Canadian Journal Mathematics*, 17:449–467, 1965.

[25] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–8, January 2000.

[26] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–6, August 2002.

[27] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, January 2000.

[28] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, 1979.

[29] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, December 1976.

[30] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[31] L. Glass. A topological theorem for nonlinear dynamics in chemical and ecological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 72(8):2856–2857, 1975.

[32] L. Glass and S. A. Kauffman. The logical analysis of continuous, nonlinear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973.

[33] K. Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. In K. Berka and L. Kreiser, editors, *Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik (vierte Auflage)*, pages 347–370. Akademie-Verlag, Berlin, 1986.

[34] J. Hanna, M. Wernig, S. Markoulaki, C. Sun, A. Meissner, J. P. Cassady, C. Beard, T. Brambrink, L. Wu, T. M. Townes, and R. Jaenisch. Treatment of sickle cell anemia mouse model with ips cells generated from autologous skin. *Science*, pages 1920–1923, December 2007.

[35] I. Harvey and T. Bossomaier. Time out of joint: attractors in asynchronous random boolean networks. In *Proceedings of the Fourth European Conference on Artificial Life (ECAL97*, pages 67–75. MIT Press, 1997.

[36] D. Hilbert and W. Ackermann. *Grundzüge der theoretischen Logik (Principles of Theoretical Logic)*. Springer-Verlag, 1928.

[37] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison Wesley, November 2000.

[38] S. Huang, G. Eichler, Y. B. Yam, and D. E. Ingber. Cell fates as high-dimensional attractor states of a complex gene regulatory network. *Physical Review Letters*, 94(12), 2005.

[39] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, October 2004.

[40] H. Iwasaki and K. Akashi. Myeloid lineage commitment from the hematopoietic stem cell. *Immunity*, 26(6):726–740, June 2007.

[41] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, March 1969.

[42] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, May 1993.

[43] S. A. Kauffman, A. S. Ribeiro, and J. Lloyd-Price. Inferring dynamical gene regulatory networks subject to noise. In *3rd Annual Rocky Mountain Bioinformatics Meeting*, Dec 9-11 2005.

[44] L. Khachiyan. A polynomial algorithm for linear programming. *Doklady Akad Nauk USSR*, 244(5):1093–1096, 1979.

[45] D. E. Knuth. *Art of Computer Programming, Volume 1: Fundamental Algorithms (3rd Edition)*. Addison-Wesley Professional, July 1997.

[46] D. E. Knuth. *Art of Computer Programming, Volume 3: Sorting and Searching (2nd Edition)*. Addison-Wesley Professional, April 1998.

[47] E. Landau. *Handbuch der Lehre von der Verteilung der Primzahlen.* Chelsea Publishing Company, 3 edition, April 1974.

[48] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. Fitzhugh, R. Funke, D. Gage, K. Harris, A. Heaford, J. Howland, L. Kann, J. Lehoczky, R. Levine, P. Mcewan, K. Mckernan, J. Meldrim, J. P. Mesirov, C. Miranda, W. Morris, J. Naylor, C. Raymond, M. Rosetti, R. Santos, A. Sheridan, C. Sougnez, N. Stange-Thomann, N. Stojanovic, A. Subramanian, D. Wyman, and J. Rogers. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, February 2001.

[49] L. László. Graph minor theory. *Bulletin of the American Mathematical Society*, 43:75–86, 2006.

[50] L. Levin. Theory of computation: how to start. *SIGACT News*, 22(1):47–56, 1991.

[51] N. Maherali, R. Sridharan, W. Xie, J. Utikal, S. Eminli, K. Arnold, M. Stadtfeld, R. Yachechko, J. Tchieu, R. Jaenisch, K. Plath, and K. Hochedlinger. Directly reprogrammed fibroblasts show global epigenetic remodeling and widespread tissue contribution. *Cell Stem Cell*, 1(1):55–70, June 2007.

[52] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1, 2006.

[53] A. Meissner, M. Wernig, and R. Jaenisch. Direct reprogramming of genetically unmodified fibroblasts into pluripotent stem cells. *Nature Biotechnology*, 25(10):1177–81, October 2007.

[54] M. I. Monastyrsky. *Topology in Molecular Biology : DNA and Proteins (Biological and Medical Physics, Biomedical Engineering).* Springer, August 2005.

[55] K. Okita, T. Ichisaka, and S. Yamanaka. Generation of germline-competent induced pluripotent stem cells. *Nature*, June 2007.

[56] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, November 1993.

[57] M. Papetti and A. I. Skoultchi. Reprogramming leukemia cells to terminal differentiation and growth arrest by rna interference of pu.1. *Molecular Cancer Research*, 5(10):1053–62, October 2007.

[58] M. C. Reed. Why is mathematical biology so hard? *Notices of the AMS*, 51(3):338–342, March 2004.

[59] A. S. Ribeiro, R. Zhu, and S. A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of Computational Biology*, 13(9):1630–1639, 2006.

[60] B. Samuelsson and C. Troein. Superpolynomial growth in the number of attractors in kauffman networks. *Physical Review Letters*, 90(9):098701, Mar 2003.

[61] W. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[62] L. A. Segel. *Modeling Dynamic Phenomena in Molecular and Cellular Biology*. Cambridge University Press, New York, NY, USA, 1984.

[63] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, February 2002.

[64] I. Shmulevich and S. A. Kauffman. Activities and sensitivities in boolean network models. *Physical Review Letters*, 93(4):048701, Jul 2004.

[65] M. Sipser. On relativization and the existence of complete sets. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 523–531, London, UK, 1982. Springer-Verlag.

[66] J. E. S. Socolar and S. A. Kauffman. Scaling in ordered and critical random boolean networks. *Physical Review Letters*, 90(6):068702, Feb 2003.

[67] E. Spaan, L. Torenvliet, and P. van Emde Boas. Nondeterminism fairness and a fundamental analogy. *Bulletin of the European Association of Theoretical Computer Science*, 37:186–193, 1989.

[68] R. Steuer. Effects of stochasticity in models of the cell cycle: from quantized cycle times to noise-induced oscillations. *Journal of Theoretical Biology*, 228(3):293–301, June 2004.

[69] G. Stolovitzky, D. Monroe, and A. Califano. Dialogue on reverse-engineering assessment and methods: the dream of high-throughput pathway inference. *Annals of the New York Academy Science*, 1115:1–22, December 2007.

[70] K. Takahashi, K. Tanabe, M. Ohnuki, M. Narita, T. Ichisaka, K. Tomoda, and S. Yamanaka. Induction of pluripotent stem cells from adult human fibroblasts by defined factors. *Cell*, 131(5):861–72, November 2007.

[71] K. Takahashi and S. Yamanaka. Induction of pluripotent stem cells from mouse embryonic and adult fibroblast cultures by defined factors. *Cell*, 126(4):663–676, August 2006.

[72] P. Turchin. *Complex Population Dynamics: a Theoretical/Empirical Synthesis*. Princeton University Press, Princeton, NJ, 2003.

[73] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265,

1936.

[74] P.B. Warren and P.R. ten Wolde. Chemical models of genetic toggle switches. *Journal of Physical Chemistry B*, 109(14):6812–6823, 2005.

[75] M. S. Waterman. Applications of combinatorics to molecular biology. In *Handbook of Combinatorics (Volume 2)*, pages 1983–2001. MIT Press, Cambridge, MA, USA, 1995.

[76] M. S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall/CRC, June 1995.

[77] M. Wernig, A. Meissner, R. Foreman, T. Brambrink, M. Ku, K. Hochedlinger, B. E. Bernstein, and R. Jaenisch. In vitro reprogramming of fibroblasts into a pluripotent es-cell-like state. *Nature*, June 2007.

[78] J. Yu, M. A. A. Vodyanik, K. Smuga-Otto, J. Antosiewicz-Bourget, J. L. L. Frane, S. Tian, J. Nie, G. A. A. Jonsdottir, V. Ruotti, R. Stewart, I. I. I. Slukvin, and J. A. A. Thomson. Induced pluripotent stem cell lines derived from human somatic cells. *Science*, November 2007.

[79] S. Zhang, M. Hayashida, T. Akutsu, W. Ching, and M. K. Ng. Algorithms for finding small attractors in boolean networks. *EURASIP Jounal of Bioinformatics Systems Biology*, 2007(1):4–4, 2007.

[80] H. Zhu, S. Huang, and P. Dhar. The next step in systems biology: simulating the temporospatial dynamics of molecular network. *Bioessays*, 26(1):68–72, January 2004.