An Algebraic Approach to Modelling the Regulation of Gene Expression

A thesis submitted to the University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2010

HOSAM ABDEL ALEEM

SCHOOL OF CHEMICAL ENGINEERING AND ANALYTICAL SCIENCE

Contents

List of Figures

Nomenclature

x, x_1, x_2, ...	Independent (mathematical) variables that can represent different biological variables such as the activation state of a protein or the concentration of a chemical species; that can be considered as inputs to a gene regulatory function. In most of this work, the values of these variables are binary or discrete multiple-valued.
y, y_1, y_2, ...	Dependent (mathematical) variables that can represent different biological variables such as the expression level of a gene; that can be considered as outputs to a gene regulatory function. In most of this work, the values of these variables are binary or discrete multiple-valued. In general, letters towards the end of the alphabet e.g. *u*, *v*, *w*, *x*, *y* and *z* represent variables.
a_0, a_1, a_2, ...	Coefficients in an equation. In most of this work, the values of these coefficients are binary or discrete multiple-valued.
d_0, d_1, d_2, ...	Truth values of a logic function. In most of this work those values are binary or discrete multiple-valued. In general, letters towards the beginning of the alphabet e.g. a *b*, *c*, *d*,.. etc. represent variables. When there are several equations in several unknowns and several coefficients, they are represented in matrix form as is common in linear algebra. In such a case vectors and matrices are denoted by bold face letters.
i, j, k, ...	Indices, i.e. running values usually ranging from 0 or 1 to some positive integer n or m. In general, letters towards the middle of the alphabet represent indices.
GF(q)	Galois Field (also known as finite field) of order q. It is an algebraic structure defined on a set with a finite number of elements q. q is either a prime or a positive power integer power of a prime.
$\oplus$	Indicates addition on a finite field.

Abstract

An Algebraic Approach to Modelling the Regulation of Gene Expression

Hosam Abdel Aleem
The University of Manchester
Doctor of Philosophy
September 2010

Biotechnology is witnessing a remarkable growth evident both in the types of new products and in the innovative new processes developed. More efficient process design, optimisation and troubleshooting can be achieved through a better understanding of the underlying biological processes inside the cell; a key one of which is the regulation of gene expression. For engineers such understanding is attained through mathematical modelling, and the most commonly used models of gene expression regulation are those based on differential equations, as they give quantitative results. However, those results are undermined by several difficulties including the large number of parameters some of which, such as kinetic constants, are difficult to determine. This prompted the development of qualitative models, most notably Boolean models, based on the assumption that biological variables are binary in nature, e.g. a gene can be on or off and a chemical species present or absent. There are situations however, where different actions take place in the cell at different threshold values of the biological variables, and hence the binary assumption no longer holds.

The purpose of this study was to develop a method for modelling gene regulatory functions where the variables can be thought of as taking more than two discrete values.

A method was developed, where, with the appropriate assumptions the biological variables can be regarded as elements of an algebraic structure known as a finite field, in which case the regulatory function can be considered as a function on such a field.

The formulation was adopted from electronic engineering, and leads to a polynomial known as the Reed-Muller expansion of the discrete function.

The model was first developed for the more familiar binary case. It was given three different algebraic interpretations each enabling the study of a different biological problem, albeit related to gene regulation.

The first interpretation is as a function on a Boolean algebra, but using the Exclusive OR (XOR) operation instead of the OR operation. The discriminating superiority of the XOR allows a more efficient determination of the gene regulatory function from the data, a problem known as reverse engineering.

The second interpretation is as a polynomial on a finite field, where analogy with the Taylor series expansion of a real valued function allowed the coefficients of the expansion to be thought of as conveying sensitivity information. Furthermore a method was devised to detect mutation in the cell by regarding the problem as detecting a fault in a digital circuit.

The third interpretation is as a transform on a discrete function space, which was demonstrated to be useful in synthetic biology design.

The method was then extended to the multiple-valued case and demonstrated with modelling the gene regulation of a well known example system, the bacteriophage lambda.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning

Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://www.campus.manchester.ac.uk/medialibrary/policies/intellectual-property.pdf), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University's policy on presentation of These

Acknowledgement

I would like to thank my supervisor Professor Ferda Mavituna for her guidance, support and patience throughout this study. I am very grateful for the financial support provided by the EPSRC and Scottish Power through a Dorothy Hodgkin Postgraduate Award, without which this work would not have been possible. I am also grateful to the University of Manchester for providing additional funding, in this respect I would like to thank both Ms. Sandra Kershaw and Ms. Samina Latif for their valuable help with the financial matters.
I am deeply indebted to Dr. David H. Green of the School of Electrical Engineering at the University of Manchester for useful discussions and for leaving me some valuable books and papers on his retirement. I am also grateful to Mr. Peter Senior for acting as my examiner for the first and second year reports, and for his valuable feedback.

I am grateful to Dr. Esther Ventura-Medina and the School of Chemical Engineering and Analytical Science for giving me the opportunity to demonstrate and facilitate on the first year undergraduate modules, an experience that I really enjoyed. I would also like to thank Dr. Dee Ann Johnson, and the Graduate Development Scheme (GraDS) team headed by Dr. Jim Boran, for giving me the opportunity to present at their workshops, and for the many workshops and training sessions I have attended with them. I am also grateful to Research Councils UK (RCUK) for the many training courses off campus that they have funded.

I am grateful to my friend and colleague Dr. Raul Monoz-Hernandez for inviting me to join him on the RCUK business plan competition together with the rest of the InLife Technologies team (Julio, Chakko and Gabby). This was an experience that I have greatly enjoyed and benefited from; we were shortlisted in the competition. I would also like to thank my colleagues and friends in room B9 in the Mill, and on the EBL sessions and in the School in general for their friendship and companionship.

During the course of this work, I suffered a major life threatening health problem for which I spent some time in hospital and later at home, this caused me great delay in completing this work. It was also a cause of great distress and anxiety. As usual, Professor Mavituna offered her unconditional support and understanding, for which I am truly grateful.
Last but most certainly not least I am grateful to my sister Dr. Eiman Aleem and my cousin Mr. Osama Kotb, who both came from overseas leaving their work and other commitments behind, to be with me during those difficult times.

Chapter 1: Introduction

1.1 Background

Modern Biotechnology can be viewed, from an engineering perspective, as the utilisation of living organisms or parts thereof for industrial purposes. As such it is the technological application of modern biological sciences such as molecular biology and genetic engineering in addition to more traditional ones such as biochemistry. Biotechnology has been witnessing a spectacular and steady growth that has been coupled with the advancement in those sciences, in particular over the past two decades. This growth is evident in the new markets in which biotechnology based products and services are being deployed and in the increase of their share in the markets in which they already exist. Several recent government and industry associations' reports reflect this fact (BioIndustry Association 2004; European Biopharmaceutical Enterprises 2007; Institute for Manufacturing 2007; Zika *et al.* 2007). Furthermore, the increase in funding for research and development both in industry and academia, and the increase in investment both private and public in biotechnology is further evidence of this trend (BioIndustry Association 2004; Lahteenmaki and Lawrence 2006). This is further supported by the increase in the number of patents awarded in the related biosciences; several times a year the journal Nature Biotechnology publishes patent applications in those sciences (Caulfield *et al.* 2006).

The industrial application of biotechnology can be classified into, products and services. Products refer to the substances produced by the organism; those vary widely in the complexity of the molecule from ethanol and simple organic acids to complicated antibiotics, enzymes and biopharmaceuticals (Atkinson and Mavituna 1991; Ratledge and Kristiansen 2006; European Biopharmaceutical Enterprises 2007) More recently, the use of renewable resources as feedstock for production processes has witnessed increased interest, for sustainability and environmental impact considerations (Herrera 2004; Werpy and Petersen 2004; Lorenz and Eck 2005; Patel and Crank 2006; Zika *et al.* 2007). One particular such product that has been the

focus of much attention and debate is Biofuels due to the strategic role of energy and food supply (Farley and Rouse 2000; Werpy and Petersen 2004; Patel and Crank 2006; Ragauskas *et al.* 2006). The organism itself can also be the product of the biotechnological process, such as yeast which is utilised extensively in the baking and brewing industries.

As for services, they entail the utilisation of biotechnological processes or products either for the production of other products that may or may not be biological in nature, or to perform a given task which may fall in any of several application domains. Applications of industrial relevance include environmental ones such as waste treatment (Atkinson and Mavituna 1991; BioIndustry Association 2004; Ratledge and Kristiansen 2006; Institute for Manufacturing 2007), and also - of special importance to engineers - biocatalysis and biotransformation (Straathof *et al.* 2002; Herrera 2004; Kaul *et al.* 2004; Werpy and Petersen 2004; Lorenz and Eck 2005; Patel and Crank 2006), due to the high selectivity of enzymes which enables a more efficient and economic production of chiral compounds. In addition to the modern applications of biotechnology, there are of course the more traditional ones such as baking, brewing and cheese making.

Oftentimes in order to produce the product or perform the task required, the metabolism of the organism has to be manipulated for example to block a consuming pathway or to enhance the flux of a desired product. This is achieved through engineering the metabolic process of the cell, a field of biosciences appropriately known as metabolic engineering. It involves manipulating the enzymatic and regulatory processes in the cell using recombinant DNA technology in order to achieve the desired changes in metabolism (Bailey 1991; Nielsen 2001). Enzymes are products of the expression of genes, regulatory processes regulate this expression, while recombinant DNA technology is used to insert or delete genes from the genome of the organism (these terms and others that appear in the remainder of this chapter will be explained later in this report). To be able to do this in a manner that achieves the desired results there is a need for understanding how those genes operate, what factors control them and how, and also what cellular processes they control.

Unlike the industrial applications outlined above, medical applications of modern biological sciences have a different focus. They are not concerned with production but with understanding, in particular the causes of disease and consequently how to cure them, for example through identifying drug targets. Many diseases can be attributed to problems in the regulation of gene expression; a prominent example is cancer which involves loss of control of the cell division cycle. The number of genes involved in cancer development is estimated to be more than eighty and the number is continually increasing with new discoveries (Vogelstein and Kinzler 2004). Other medical applications include tissue culture and the related area of stem cell research, which are intimately related to the regulation of gene expression for cell differentiation (Gilbert 2000). In addition there are many applications in the other life sciences for example in agriculture related to crops and in breeding of farm animals among many others.

One of the underlying commonalities among these different applications, whether industrial, medical or other life sciences, is that they all rely on the understanding of gene expression and how it is regulated. Such understanding is attained through experiments in molecular biology and other related modern biological sciences. The development of increasingly advanced analytical technologies coupled with sophisticated information technology (IT) and software functionality, resulted in the generation of massive amounts of data from these experiments. Indeed, the proliferation in the amount and type of information related to gene expression and the different functions in a living cell over the last decade has been overwhelming. These have been collectively referred to as omics and include information related to the genes (genome), the transcription of these genes (transcriptome), the proteins resulting from their translation (proteome) and the metabolic activities mediated by some of those proteins (metabolome) among other "omes". These different layers of functionality interact to yield the behaviour exhibited by the cell, whether normal or anomalous.

A collection of data alone is not sufficient to reveal the underlying causes. In order to be able to understand the complex interactions involved in the regulation of gene expression and how the different functions are carried out, a systems approach is needed that explicitly takes into account such interactions and integrates this data

(Wolkenhauer *et al.* 2003). Indeed, one of the major endeavours in a systems based approach in general is to attempt to understand the interactions of the different components of a system and the functionality that emerges from such interactions (Doebelin 1980). The main tool utilised in investigating this problem is mathematical modelling, which is the topic of this work, namely the mathematical modelling of gene expression regulation, in particular using discrete mathematics.

1.2 Mathematical modelling of gene expression regulation

Mathematical modelling essentially matches a method to a problem; it invents neither. For a given problem, there are several modelling methods each employing different mathematical formalisms or variations of a given one. Thus for a new method to be accepted it has to provide some benefits on existing ones such as computational efficiency, different insight into and interpretation of the problem or the capability to investigate new functionality that is not easily achievable under the existing methods. In this work we introduce a method that provides the last two benefits. In particular the method we use is based on abstract algebraic concepts as will be detailed in later chapters, and the problem we attack is that of the regulation of gene expression.

The expression of a gene is normally controlled by several factors, some may be internal to the cell such as the growth stage the cell is at or the division phase of the cell cycle, where in either case different functions are required by the cell. Other factors may be external to the cell, for example the available nutrients in the surrounding environment or signals from other cells. Hence from a mathematical point of view we can represent gene expression as a function of the different factors that affect it.

As with any modelling task, there are several mathematical approaches to formulate this functional relationship that depend on what we want to study, the data available, the level of detail desired, and indeed the purpose of the modelling exercise to begin with. By and large the most common modelling approach uses differential equations where the different variables take continuous values; however, it is not the only approach.

From a systems point of view, there is more than one way to classify a system. One classification is into continuous and discrete where the terms refer to the variables involved, whether they take continuous or discrete values. In a more abstract sense those variables describe the states of the system, and would correspondingly belong to either a continuous or a discrete set. Other classification include stochastic versus deterministic, static versus dynamic among others.

Differential equation models are easy to understand conceptually as they represent rates of change of some variables with respect to time and how they relate to other variables, resulting in a system of simultaneous differential equations. Because of their dependence on the time element, they can use numerical data representing a time course, available from experiments and can also generate similar data. Thus in essence the main advantage of differential equation models is that they can produce quantitative data. However, there are major problems with those models, for example most tend to use linear time invariant representation of systems producing linear constant coefficient ordinary differential equations. Those ignore nonlinearities of functions, time dependence of parameters and spatial distribution of variables, when such effects are taken into account they lead to exceedingly complicated equations potentially non-linear time-varying partial differential equation that are difficult to solve even for simple special cases. Furthermore, in formulating the model on a molecular level, often choices have to be made concerning which molecules and which molecular mechanisms to include in the model, for example delays due to transport phenomena are often ignored. Differential equations are sometimes based on kinetic models of reactions and as is well known kinetic parameters and also affinity constants are difficult to measure, hence in many cases they are estimated from the data or their values just assumed. Perhaps more importantly is that in some cases the assumptions on which the differential equation paradigm is based might not be valid to begin with. This is the case when there is a small number of molecules present in the cell, in which case the assumption of a continuous change in their concentration might not be valid (Vilar *et al.* 2003). Hence there is a need for other types of models that do not involve the complexities and the uncertainties outlined above, but need only capture the qualitative behaviour of a system. Clearly this will lose the quantitative power of differential equation models but gain simplicity in return.

One common qualitative approach is that based on the Boolean modelling formalism, which assumes that all variables are binary in nature, i.e. they can take only one of two values. Thus a gene can be either on or off instead of having different levels of expression, a protein can either be activated or de-activated instead of having different states of activation, similarly an effector molecule can be either present or absent rather than having different concentration values. Whilst this is an extreme case of the continuous variation, limiting it to the two extremes of its range of values, it does have conceptual, experimental and even mathematical justifications. It should be noted that the Boolean approach is used in logic design of electronic circuits; we will thus make use of the wealth of knowledge and techniques in this area and apply it to the modelling of gene expression regulation.

One of the main drawbacks of the Boolean approach however, is that it restricts the number of values of a variable to two only, a situation that is not always realistic in the context of the regulation of gene expression. Hence in this work we will present a method that is easily extendable to the multiple-valued (yet discrete) case, but more importantly provides additional insight into the regulation problem and useful functionality that is not easily attainable with the usual Boolean approach.

1.3 Aims and objectives of this work

To recap the discussion above, we outline the following:

- The goal of this work is to produce a method for modelling gene regulatory functions using a discrete multiple-valued mathematical representation.
- The motivation behind this work is the desire to optimise biotechnological processes through attaining a better understanding of the regulation of gene expression underlying them.
- The specific objectives that contribute towards achieving this goal are to
    - Introduce a method for the qualitative mathematical modelling of binary gene regulatory functions.
    - Investigate the new biological perspective of the mathematical formulation provided by this method and the potential new biological problems that can be studied using it.

    - Extend the method to the multiple-valued case.
    - Apply the method to an existing problem to demonstrate its utility.
- The philosophical approach underlying this work is to use abstraction in order to separate the details of a problem from its core concepts, thus revealing the commonality between it and similar problems in other application domains; hence allowing the use of existing methods from such domains.
- The pedagogical approach (so to speak) is two-fold; to start with concepts the reader is already familiar with from which to abstract to the new concepts, and to start with simple special cases from which to generalise to the more general case.
- The outcome of this work is a method and not a model.

1.4 Organisation of this report

Writing a report that spans more than one discipline is not an easy task, in particular choosing the appropriate level of detail. Inevitably some readers will find the treatment too detailed while others will find it lacking in detail; striking a balance between the two is always a challenge for someone writing for readers from different backgrounds. Indeed in the preface of their introductory book on gene expression Ptashne and Gann (2002) describe this dilemma by stating "*We face the strain of deciding where details illuminate or obscure the main points*". This work is no exception, as it spans both biology and mathematics albeit from an engineering perspective. Since this work is presented to an engineering school the reader is more likely to be familiar with mathematics than with biology. It was decided to try to give as much background in biology as required and at an elementary level, unfortunately risking oversimplifying or stating the obvious in some instances. On the other hand it was also decided to avoid fine details that would not help in the development of the work, effectively "*obscuring*" rather than "*illuminating*" the argument. With regard to mathematics, some of it will be familiar to all engineers such as differential equations and those will be treated rather concisely. Other mathematical tools might not be familiar to some engineers and those will be introduced in more detail, but only as necessary to elucidate the concepts and not more. This work is rather mathematical in nature and hence abstract. It can be considered to fall in the realm of theoretical

biology, which is essentially the theoretical study of biology often from a mathematical perspective, in some sense similar to theoretical physics (Westerhoff 2007). Research in biology is normally conducted, and knowledge generated using observations and experiments. Theoretical biology on the other hand uses logic and reasoning, both tools of mathematics, and indeed mathematics itself, to produce models and theories that can interpret or predict observations or generate hypotheses that can be tested experimentally.

The purpose of this study is to develop a method for the mathematical modelling of the regulation of gene expression based on discrete mathematics, in particular adopting techniques and ideas from logic design of electronic circuits. Towards this end we chart the following course.

In this chapter, chapter one, we have started by providing the motivation for this work. As engineers our first motivation was applied, i.e. the industrial application of biotechnology, and we have also touched on other application domains mainly the medical one. This has led us to the conclusion that all the applications irrespective of their nature require an understanding of the regulation of gene expression. Hence the applied has led us to the basic science, which we indicated generates large amounts of data. This then led us to the convenience or even the necessity of utilising mathematics in order to understand the interactions between the different processes generating the data. After briefly discussing differential equation models, the most common modelling approach, and outlining its benefits and its shortcomings, we proposed discrete models in particular an approach similar to that used in logic design. Hence the logical progression of the argument thus far is as follows: Maximising applied benefit requires understanding the basic science which requires the use of mathematical models and among those discrete models have potential benefits that have not been adequately explored.

Chapter two is about what we want to model which is the regulation of gene expression. We first introduce some fundamental concepts from molecular biology that we then use as a foundation to build upon the main topic namely gene expression and how it is regulated. In chapter two we will explain many of the terms encountered in section 1.1 above.

In chapter three we address the modelling of gene expression regulation. In the context of this work, mathematical modelling can be considered as the research method, i.e. the tool modellers use to understand gene expression regulation (the ultimate goal). Hence it is important to understand some of the implicit assumptions associated with mathematical modelling in order to be able to assess the validity of a model. In particular we will discuss some notions related to modelling, such as its purpose, tools and limitations and some underlying epistemological issues. To use engineering analogy, modelling is essentially a process whose inputs are the observations and the knowledge of the underlying system being modelled, and whose output is the model (figure 1-1). From such a perspective the model produced will not only depend on the data and knowledge available about the system but will also determined by the limitations of the mathematical machinery used, in a similar way as the output of an industrial process is limited by its capability.

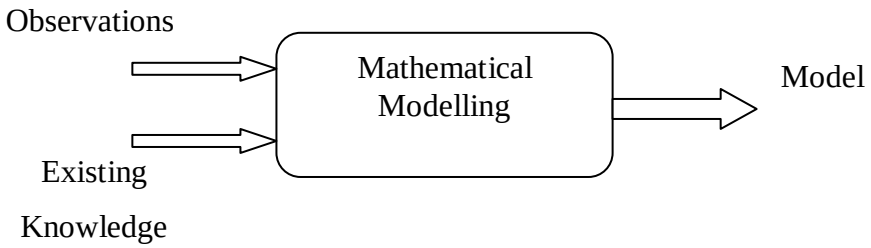


Figure 1-1: Mathematical modelling as an input/output process.

After this introduction we survey some of the common mathematical methods used in modelling the regulation of gene expression. Those are mainly based on differential equations, but we will also discuss Boolean models found in the literature and their limitations, which will lead us to suggest our own method based on concepts from abstract algebra. Chapter three is an elaboration on section 3.2 above.

In chapter four we will thus introduce some fundamental concepts from algebra which will be used in developing our method. Those are abstractions and generalisations of some of the more familiar concepts. There are two main algebraic notions that we will want to take forward from that chapter, namely finite fields and linear vector spaces.

Chapters five and six constitute the main contribution of this work, namely representing discrete gene regulatory functions on finite fields and function spaces, thus they build on the material in chapter four. In chapter five we will address the binary case where we will represent it first as a function on the usual Boolean algebra but with a different interpretation, and then as a polynomial on a finite field. We will suggest biological problems that can be studied by those techniques, namely reverse engineering of gene regulatory functions, and mutation detection. In chapter six we will represent the binary case as a transform on a function space, then generalise it to the multiple-valued case. In both we will suggest how such a method can be used to design biological systems in what is currently known as synthetic biology.

In chapter seven we apply the method developed in chapter five and six to a biological example, namely the phage lambda in the bacterium *Escherichia coli*, which is used as a model system for studying gene expression regulation. It is important to remember that in this work we develop a method rather than produce a model. Hence the example systems are used for demonstrating the method rather than for their own right. Simple well studied example cases are used in order not to mask the method by the complexity of the system it is applied to.

In the final chapter we will summarise the whole development, point out to the limitations of the method and to avenues of research that it opens up both within the context of this work and in the wider context.

Each chapter will start with a roadmap of what is going to be covered in it, and will end with a short summary and when appropriate conclusion that highlights its main points and links it to the chapter that follows it in what is hoped to be a logical succession of ideas forming a linked chain.

It is acknowledged that this work might not fit in the standard pattern of presentation whereby it would be organised in standard chapter titles such as “introduction, literature review, method, results, discussion”. However, all those elements are covered here albeit in a different guise. Figure 1-2 depicts a mapping between the chapters of this work and the standard topics.

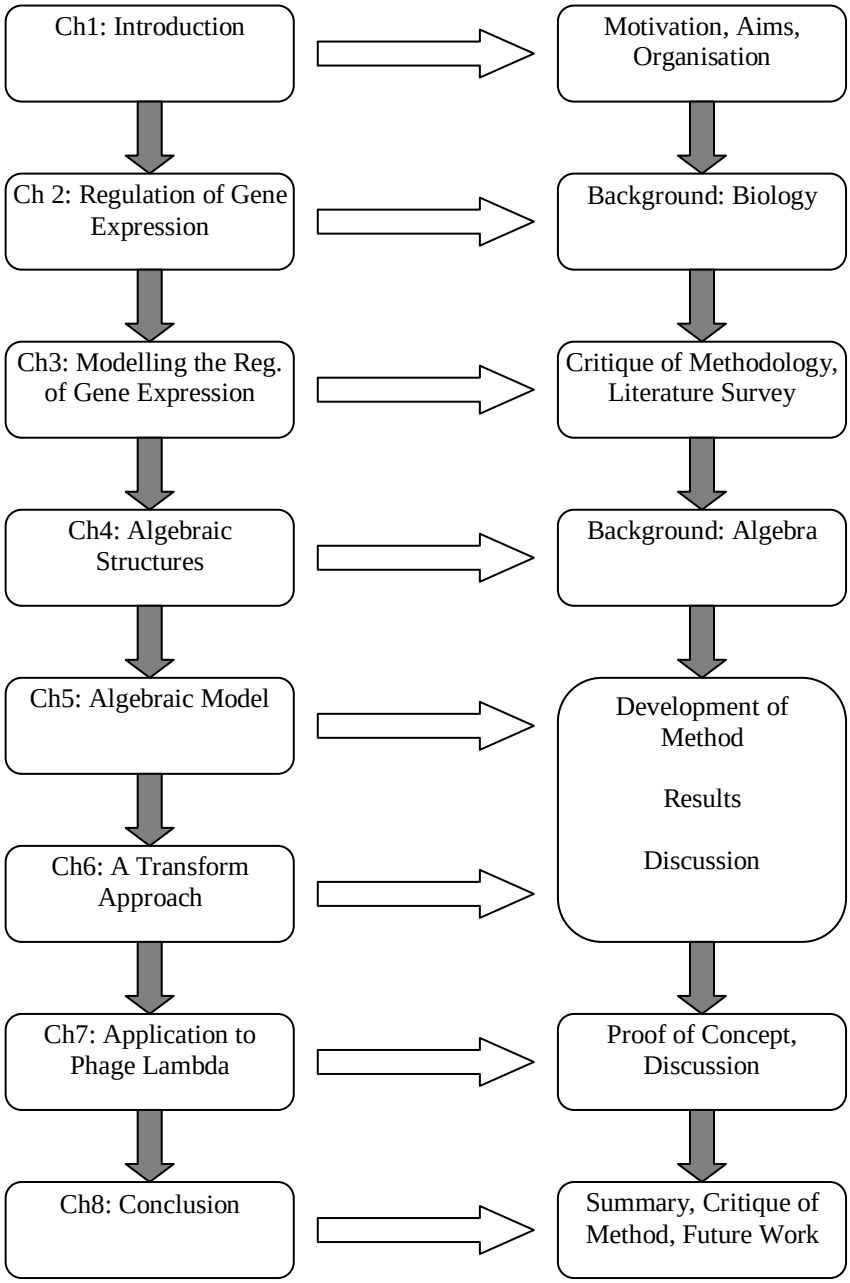


Figure 1-2: Correspondence between the contents of this work and the standard topics.

Vertical arrows do not necessarily indicate dependence.

1.5 Summary

This chapter provided motivation for the choice of problem and for the choice of method for solving this problem. In addition the problem itself was defined namely modelling the regulation of gene expression using discrete mathematics. The aims and objectives were outlined and the organisation of the report was presented.

Chapter 2: The Regulation of Gene Expression

2.1 Introduction

This report is about mathematical modelling of the regulation of gene expression. As is common in engineering modelling tasks, one needs to acquire some understanding of the main components of a system and how they interact in order to be able to capture them in a model. The purpose of this chapter is thus to present the main concepts and principles underlying the regulation of gene expression. We do so by addressing three main questions namely, what is gene expression? Why does the cell need to regulate it? And how does it do that? In order to answer these questions we first need to briefly introduce some basic concepts from cell biology. This will also help us to set the ground for the rest of this work and outline the terminology used.
The material will be presented in a way that we hope is amenable to engineers, through making analogies to concepts from chemical engineering, in particular process control.

2.2 Some basic concepts from cell biology

A cell is the building block or smallest unit of any living organism. Such organisms may consist of one or more cells, up to many millions (Alberts *et al.* 2004). Survival is arguably the ultimate goal of all living beings, thus a cell must have the ability to perform the necessary functions required for its own survival, that of the organism it is part of (if any), and of its species as a whole. A brief and structured look at the cellular functions associated with each of these levels of survival is presented below.

Functions that intrinsically relate to the cell's own survival include synthesising (and/or utilising) the necessary molecules and producing the energy required for this and other biological activities. Such activities take place inside the cell and hence have to be coordinated both structurally (in terms of space) and functionally (in terms of time, i.e. issues of precedence and concurrency). To achieve this, a cell has different compartments (known as organelles) in which different functions take place;

of special importance among those compartments is the cell nucleus which houses its genetic code (DNA). Such cells are termed Eukaryotes and are mainly found in multi-cellular organisms. On the other hand, some of the organisms made up of a single cell do not have this intricate spatial segregation; in particular they lack the cell nucleus and are termed Prokaryotes (figure 2-1). In such a case the DNA is present along with other molecules in the cell body without being segregated. Indeed, the fact that Prokaryotes are made-up of a single cell means that this cell has to perform all the functions that would otherwise be distributed over many cells. This requires higher efficiency and agility on part of the organism. While all bacteria are Prokaryotes, not all single cell organisms are, for example yeast which consists of a single cell is a Eukaryote.

Although we will occasionally mention Eukaryotes, our focus here will be on Prokaryotes due to several reasons. Firstly, the regulation of gene expression is much better studied and understood in bacteria (Prokaryotes) than in Eukaryotes and hence Prokaryotes will provide a more reliable test bed for our model. Secondly, due to the industrial relevance of bacteria, and finally, for pragmatic reasons and that is the simpler nature of Prokaryotes. Since the purpose of this work is to develop a mathematical modelling method rather than produce a particular model, using a simple and familiar system as a test case to model will avoid masking the merits of the method in the details of what is being modelled.

Functions of the cell that relate to the survival of the organism of which it is part, include the ability to communicate with other cells within the organism. This communication causes the cells - among other things - to aggregate into tissue that forms organs which is particularly important in the developmental stages of the organism (Gilbert 2000). This communication is also important when the organism is under threat from external agents such as pathogens.

To clarify the matter of development, which is essentially the formation of the body of the organism, consider a building say an apartment block. The structural units, those that carry the load of the building are the columns and beams and are made of concrete or steel. Walls are made of bricks or panels, windows from glass, floors from wood or tiles and so forth. The point is that the different functional parts of the

building are made from units that are suitable for that function. The situation is similar for a multi-cellular organism, for example in the human body different organs are made from cells that are suitable for the function of that organ, e.g. the liver, muscle or blood vessel each has a different type of cell. Given that the whole human body starts from one cell which is the fertilised egg, how do the different cell types emerge? This process is known as differentiation and involves the specialisation of the cells to different functions. Differentiation is also related to and is part of another developmental process known as morphogenesis which can be viewed as the emergence of the structure and form of the organism (Gilbert 2000).

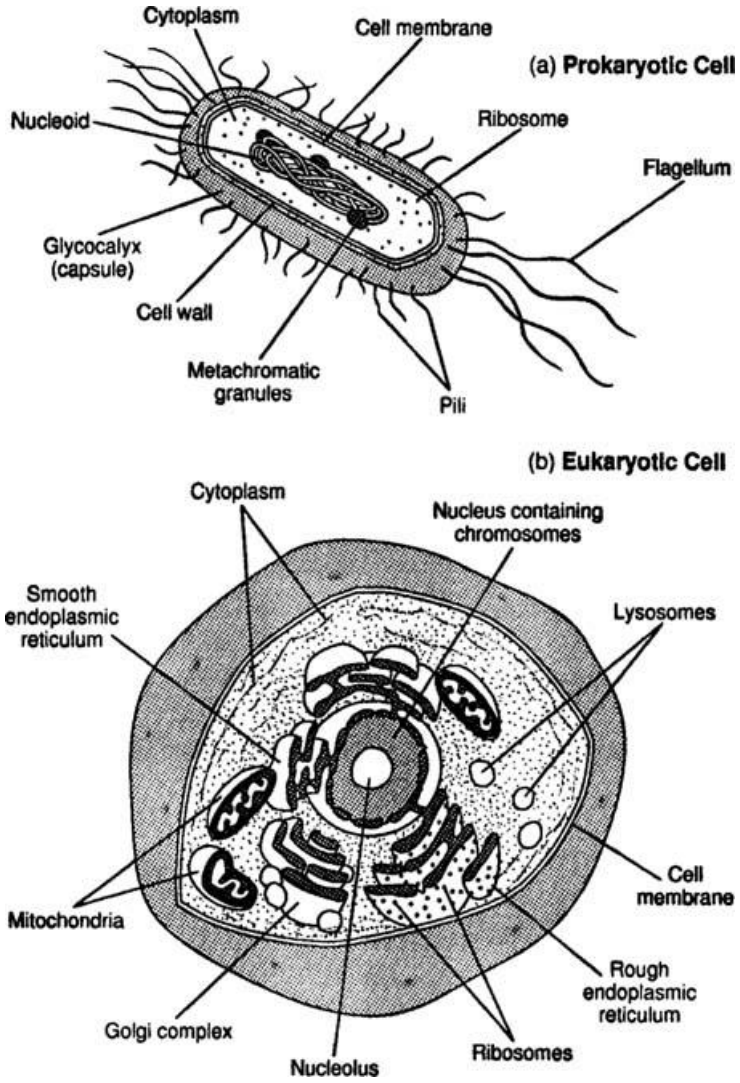


Figure 2-1: a - Prokaryotic cell, b - Eukaryotic cell.

(Source: en.wikipedia.org, under the Creative Common Attribution Share License.)

Morphogenesis is one of the major applications of the regulation of gene expression. It is worth noting that a stem cell, which is frequently discussed in the media, is a cell that can differentiate into any cell type and hence is of great importance in what is known as regenerative medicine.

Finally, for the functions of a cell that relate to the survival of the species as a whole, the most important is the ability to reproduce. For a single cellular organism, this essentially means the ability of the cell to replicate itself, i.e. to divide. The new cells must have all the molecules needed or that may be needed for their correct functioning and survival. Achieving this is of course impractical as some of those molecules may never be used during the lifetime of the cell, such as in the case of the response to certain environmental conditions like stress or starvation, or an attack by another organism that may never occur. Thus rather than replicate every molecule that may potentially be needed, it is more practical to replicate the ability to synthesise such molecules, i.e. a form of blueprint. This blueprint is the Deoxyribonucleic Acid or DNA molecule of the cell, and it contains all the information necessary for survival of the organism, in a coded form often referred to as its genetic code (Alberts *et al.* 2004).

2.2.1 The genetic code

The DNA molecule is a polymer that consists of a string of units known as nucleotides. A nucleotide is formed of three main chemical components namely, a pentose sugar known as Deoxyribose sugar, a Phosphate group and an organic base (figure 2-2).

There are four types of bases known as Adenine, Guanine, Thymine and Cytosine, denoted by A, G, T and C respectively, leading to four corresponding types of nucleotides. The DNA molecule has a particular double stranded structure famously known as the “double helix” (figure 2-3). Within the limits imposed by this structure, only certain combinations of bases can interact with high affinity, these are A-T and G-C and are thus referred to as base pairs (figure 2-3) (Nelson and Cox 2000).

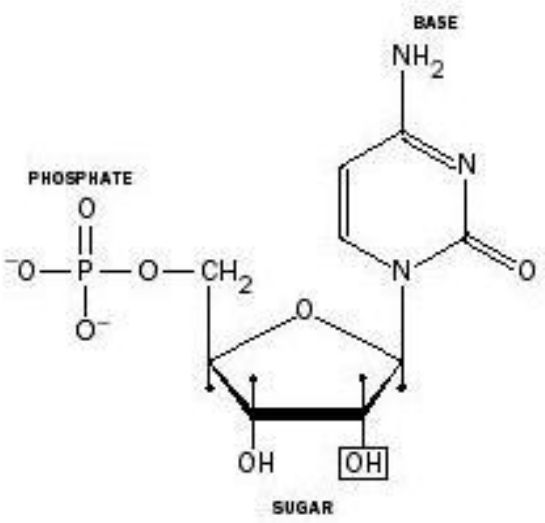


Figure 2-2: A nucleotide

(Source: en.wikipedia.org, under the Creative Common Attribution Share License.)

One of the fundamental molecules that the DNA codes for is a protein which is of utmost importance for cellular activities. Proteins have both structural and functional roles in the cell. The former involves forming part of some of the structural units in the cell such as the cell wall. The functional role of proteins is varied and includes acting as receptors on the cell membrane to detect external signals and relay them to the appropriate location in the cell, and acting as channels that allow molecules in and out of the cell. Proteins also have a crucial regulatory role within the cell as explained below.

The backbone of a protein is a chain of amino acids known as a polypeptide, and it ranges in length from a few hundred amino acids for small proteins to a few thousands for large ones (Alberts *et al.* 2004). The chain folds in different conformations depending on the amino acids present and other factors such as any other molecules attached to the chain. There are twenty different types of amino acids in the cell. Given that there are only four different types of nucleotides, the minimum number of nucleotides needed to code for an amino acid is three. Indeed three nucleotides taken together are known as a codon and they code for one amino acid. The number three comes from the fact that three nucleotides, each being one of four possible types gives four to the power three, i.e. $4^3 = 64$ (sixty four) different code words. This means that some of the twenty amino acids will have more than one code,

known as synonyms, providing robustness against errors in a sense similar to the codes used in communication engineering (May *et al.* 2004). Also some of the additional code words, called stop codons, are used to indicate the end of transcription.

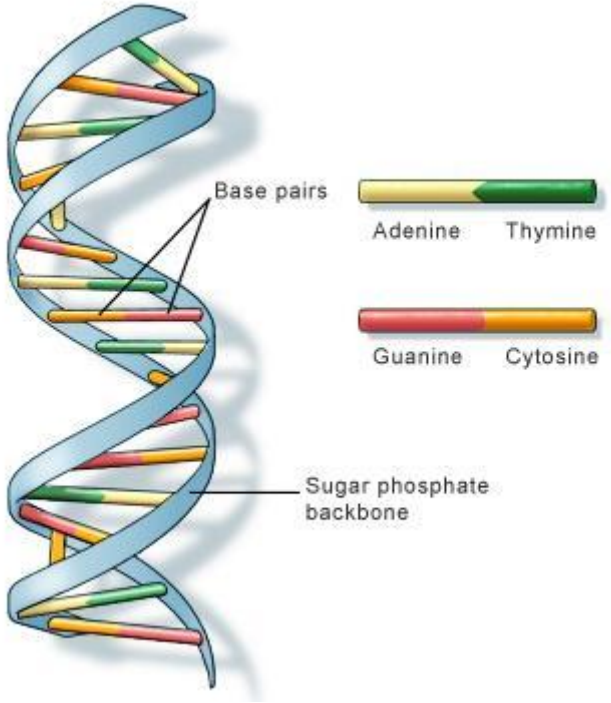


Figure 2-3: The double helix of the DNA molecule.

(Source: en.wikipedia.org, under the Creative Common Attribution Share License.)

The sequence of nucleotides on a DNA molecule that codes for one protein is known as a gene. More precisely, genes that code for a protein are known as structural genes, and that is because not all genes code for proteins as some code for other molecules such as the Ribonucleic Acid (explained below) which forms part of other functional units in the cell. The total number of nucleotides in a DNA molecule, which form the genome of the organism, can range from a few thousands for some bacteria to billions for humans and other primates corresponding to up to tens of thousands of genes (Alberts *et al.* 2004; Davidson 2006).

2.3 What is gene expression?

In general, gene expression refers to the process by which the information in the DNA is transformed into cellular function. This function is often but not always

carried out by a protein, hence the process often culminates in the synthesis of a protein. Towards this end a series of steps takes place that starts with the transcription of the gene from the DNA into another form known as the RNA, followed by the translation of the RNA into a protein that is then assembled and processed in the necessary way to carry out its ultimate function. This flow of information is commonly known as the central dogma of molecular biology and stipulates that information flows in one direction only, i.e. DNA → RNA → Protein (figure 2-4).

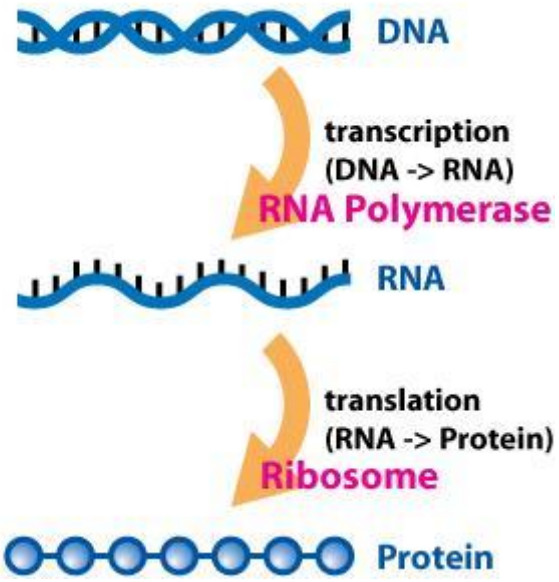


Figure 2-4: Gene expression – the central dogma of molecular biology.
(Source: en.wikipedia.org, under the Creative Common Attribution Share License.)

RNA which stands for Ribonucleic Acid is also a nucleic acid and like DNA it consists of a string of nucleotides, where now the Deoxyribose sugar is replaced by Ribose sugar, and the base Thymine is replaced by another base, Uracil (U).

The transcription step is necessary because DNA being the blueprint has to be kept intact for future use. Transcription effectively makes a copy of the blueprint for use in the "production run" of a protein. The transcribed RNA molecule on the other hand is mainly used to take the message from the DNA to the place in the cell where it will be translated into an amino acid chain, hence is referred to as the messenger RNA (mRNA). Transcription is carried out by a very important enzyme known as RNA polymerase together with the aid of other molecules. The translation machinery in the

cell is known as the ribosome and consists of a large complex of proteins and another RNA molecule known as ribosomal RNA (or rRNA), (figure 2-4). The translation process uses a third type of RNA known as transfer RNA (or tRNA) which transfers the necessary amino acids to the ribosome to add to the growing polypeptide chain.

Because it is not needed after relaying the message, the mRNA molecule does not need to be particularly stable; in fact this is actually desirable in order to get rid of un-used mRNA molecules, and is one of the tools used by the cell to regulate the rate at which a gene is expressed as explained below. The reduced stability of the mRNA molecule compared to that of the DNA molecule is achieved by the structural difference between the two, (table 2-1). On the other hand, because the DNA molecule carries all the information needed for the preservation of the life of the cell and the species, it has to be stable as it hands down this information from one generation to another through cell division. This stability is achieved by the rigid specifity of base pairing imposed by the structure of the molecule, together with the components involved in that structure namely the type of sugar and bases involved.

Table 2-1: Comparison of DNA and RNA molecules.

Comparison criteria	DNA	RNA
Sugar	Deoxyribose	Ribose
Bases	A, C, G & T	A, C, G & U
Structure	Double-stranded	Single-stranded
Stability	High	Lower

2.4 Why does a cell need to regulate the expression of its genes?

As mentioned earlier the cell might not need all the proteins it can produce or the functions they perform all the time. Furthermore, for a multi-cellular organism different organs will have different types of cells that will express different genes depending on the function of that organ. Thus the cell must have a means by which to decide when to express a certain gene and when not to, and also the levels to which it needs to express it. This is achieved through the regulation of gene expression. Indeed

this flexibility leads to one of the remarkable features of a living organism, namely its ability to adapt to its environment and to its own varying needs. Consequently the decision of which genes to express is based on several factors, some are internal to the cell while others are external to it, whether from the environment as in the case of single cell organisms, or from other parts of the body for multi-cellular ones. Those two classes of factors are briefly described below.

2.4.1 Response to internal cell requirements

Different molecules are needed by the cell at different points in its lifetime. For example for bacteria in a bioreactor, different functions are performed during the growth phase than the stationary phase. Another example is with the cell division cycle which comprises several phases, each with its own function and hence gene expressions. Even when the same molecules are needed for different biological processes, their quantities might vary with time and need. On the other hand, for multi-cellular organism, for example humans, different cells are specialised to perform different functions yet all have the same DNA. Each function may require a different set of proteins; hence such cells need to be able to switch on only the genes that express the required proteins while switching off the rest of the genes (Alberts *et al.* 2004; Davidson 2006). Furthermore, changes in the genes expressed can be triggered by events intrinsic to the cell such as in response to errors in DNA replication or DNA damage, where certain molecules are required to fix the damage.

2.4.2 Response to external signals

Changes in the environment surrounding the cell, especially extreme conditions such as starvation or heat shock for bacteria, cause changes in gene expression both qualitatively (which genes are expressed) and quantitatively (the level to which they are expressed). Less drastic changes in environmental conditions can also be a cause of change in gene expression, such as the change in the type of nutrient in the environment. Similarly for a multi-cellular organism signals from other parts of the organism can trigger changes in gene expression such as in response to hormones or to chemical cues causing cell differentiation during development as explained earlier.

2.5 How does a cell regulate the expression of its genes?

Having identified the need to regulate gene expression, we now consider how the cell effects this regulation. Similar to an industrial process, different conditions may lead to certain decisions being taken by the cell to switch relevant genes on or off in a series of events. Regulation can take place at any of the different levels of gene expression, from the initiation of transcription to the degradation of proteins which are usually the final product of gene expression (figure 2-4). Clearly it would be more efficient to control expression at its inception, i.e. at the level of transcription initiation as it means that no energy is wasted in transcription or translation before the mRNA molecule (the product of transcription) or the protein (the product of translation) is degraded. However, as outlined earlier a condition may occur that necessitates halting the expression of some genes that is already in progress, or expressing others in response to the condition. Depending on such a condition the cell can employ either global controls or local ones. Global controls act on most genes at the same time, as for example in the case of extreme environmental conditions that may require the cell to halt several processes at once and invoke an emergency response, similar to a shutdown system in a process plant. Local controls on the other hand, act only on those genes involved in the function to be regulated, similar to a control loop in a unit operation of a process plant. The ability to exercise global control means that the cell can override local controls when necessary. It should also be noted that the response to changing conditions may cause some genes to increase their expression levels and others to decrease them, similar to direct and reverse acting control in industrial processes. The different levels of regulation are briefly outlined below without the details of their molecular mechanisms; those will be discussed further when dealing with particular applications in later chapters.

2.6 Control of transcription

In order for the transcription of a gene to start, the enzyme RNA polymerase which performs it needs to identify the location at which to start transcription and the direction in which to proceed. This information is indicated by a region on the DNA molecule upstream of the gene known as the promoter. In addition, it needs an

indication of the end of transcription, this is provided by a stop codon as discussed above.

The question that arises then is how does RNA polymerase recognise the promoter? An ideal promoter has a certain pattern of nucleotides termed the consensus sequence, which RNA polymerase recognises, binds to and starts the transcription process. In general, the promoter of a gene does not have the exact consensus sequence but will deviate from it. The level of transcription will depend on how close the promoter sequence is to the consensus one (Ptashne and Gann 2002). If the difference between them is large, a protein known as a transcription factor will be needed to facilitate the binding of RNA polymerase to the promoter (Ptashne and Gann 2002). Other transcription factors may also be involved in the transcription process to regulate expression resulting in either gene activation which increases the transcription rate or alternatively gene repression which decreases it (Wagner 2000). The transcription factors, being proteins, are products of other genes hence leading to genes regulating other genes which may feedback to the original ones. The result is an interconnected network of genes with feedback and feed forward interactions (Thieffry *et al.* 1998; Davidson 2006). A regulatory protein such as a transcription factor often needs a small molecule to activate it, which binds to some domain of the protein. Such molecules are known as effector molecules, and often carry information about the controlling condition (Alberts *et al.* 2004).

In summary, the changes in the different conditions affecting the cell are relayed to the transcription machinery through a cascade of signalling molecules ending with the effector molecule which binds to the transcription factor. This may then either increase the rate of expression of the gene or decrease it.

The normal un-regulated state of a gene is called its basal state, and gene regulation would then modulate this state in response to the appropriate conditions. For example a gene that is normally on would be switched off when the relevant condition occurs. Certain genes are on all the time independent of any conditions, and are referred to as being constitutively on. Examples of those are the so called housekeeping genes which are necessary for the key activities of the cell such as energy production.

For the sake of completeness we mention that transcription takes place in three stages, namely initiation, elongation and termination. Initiation involves starting the transcription process through recruiting the different molecules as described above. Elongation involves the addition of the different nucleotides to the growing RNA molecule. Finally termination releases the synthesised RNA molecule.

2.7 The *lac* operon

We now give an example of transcription regulation that clarifies some of the concepts outlined above such as transcription factor, activation, repression, constitutive, regulated, effector molecule and global and local controls. We illustrate this by a simplified presentation of the *lac* operon in the bacterium *E. coli*.

In some cases, a bacterium needs to coordinate the regulation of several genes together, for example when the products of those genes are involved in some metabolic function. In such a case the concept of an operon is employed, which is essentially a group of genes that are transcribed and regulated together. An operon consists of two functional regions that may be physically interspersed or overlapping on the DNA molecule. One region contains the structural genes, i.e. those that code the proteins contributing to the metabolic function. The other region is a regulatory region that controls the expression of the structural genes, and hence contains their promoter and may also contain other genes that produce transcription factors that control the structural genes. There are many operons in *E. coli* and they can contain as little as two or as many as twenty structural genes (Nelson and Cox 2000).

Bacteria prefer glucose as their energy source because of its relatively high potential energy. In addition glucose is also a precursor for many metabolic pathways synthesising different types of biomolecules. A precursor in this sense is similar to the feedstock in a chemical process. Glucose has a straightforward metabolic pathway to utilise it known as glycolysis. Other sugars have to be transformed into glucose or to one of its derivatives before they can be utilised by the bacteria, hence consuming energy in this conversion. Thus when the medium contains several sugars including glucose, all sugar metabolising pathways other than glycolysis have to be inhibited, i.e. their genes switched off (Nelson and Cox 2000). On the other hand when glucose

is not present but another sugar is, then the bacteria have to be able to switch on the pathway to metabolise this sugar. One of the well studied cases in *E. coli* is that of the sugar lactose where regulation is achieved through the *lac* operon (figure 2-5).

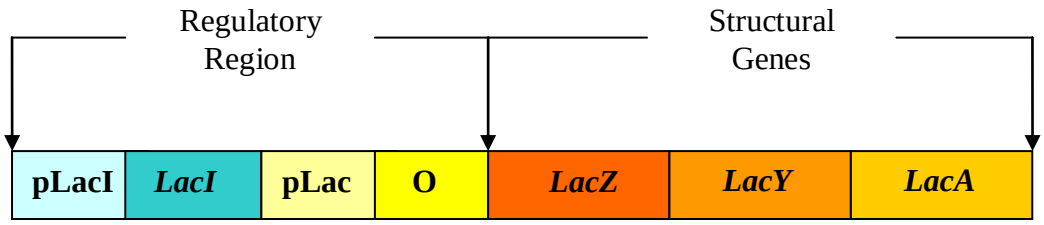


Figure 2-5: Structure of the *lac* operon.

2.7.1 Structure of the *lac* operon

Like other operons, the *lac* operon consists of a regulatory region and the structural genes. The regulatory region contains a gene called *LacI* that produces a protein called the "Lac repressor" which acts as a transcription factor. When this protein binds to the operator region on the DNA molecule it represses the structural genes. The regulatory region also contains the promoter of the *LacI* gene, and the promoter of the structural genes (all of which are regulated by a single promoter due to the nature of an operon), in addition to the operator region just mentioned. The second region of the operon contains the structural genes which code the proteins necessary for the utilisation of lactose in the absence of glucose. Note that it is a convention in the context of the *lac* operon not to consider the *LacI* gene as a structural gene even though it codes for a protein. Structurally, the *lac* operon consists of the following units on the DNA molecule as depicted in figure 2-5.

Regulatory region

- **pLacI**: Promoter for the regulatory gene *LacI*.
- ***LacI***: Regulatory gene that encodes a transcription factor known as the "Lac repressor", which as the name indicates, is a repressor protein.
- **pLac**: Promoter for the structural genes.
- **O**: Operator, a region on the DNA to which the repressor protein binds to repress the transcription of the structural genes.

There is another part of the regulatory region that is functionally distinct but structurally overlapping with the promoter pLac, and hence not shown in the figure. This is a site on the DNA molecule where an activator protein binds to the DNA molecule as explained later. Note that by convention, the names of genes are italicised.

Structural genes

- ***LacZ***: Gene encoding the enzyme β-galactosidase which cleaves lactose to produce glucose and galactose for further metabolism.
- ***LacY***: Gene encoding the enzyme galactoside permease which transports lactose into the cell from the surrounding medium.
- ***LacA***: Gene encoding the enzyme galactoside transacetylase which takes part in lactose metabolism.

2.7.2 Operation of the *lac* operon

The gene *LacI* is constitutively expressed, i.e. it is expressed all the time and its product protein, the "Lac repressor" is thus present in the cell all the time. When there is no lactose in the medium, Lac repressor binds to the operator region which overlaps the promoter of the operon (overlap not shown in figure 2-5), preventing RNA polymerase from starting transcription, hence none of the structural genes will be expressed.

Normally there is a very small amount of the enzymes β-galactosidase and galactoside permease in the cell, due to the basal expression level of the operon. Hence when lactose is present in the medium (and no glucose is present), a small amount permeates into the cell and is isomerised to allolactose. Allolactose acts as an effector molecule to the transcription factor protein Lac repressor, it binds to it and prevents it from binding to the operator region. Hence RNA polymerase can bind to the promoter of the structural genes and start transcribing the genes producing the enzymes β-galactosidase and galactoside permease allowing more lactose into the cell and lactose metabolism carries on. In this role allolactose is known as an inducer of the operon.

Glucose is the preferred energy source for *E. coli*, hence when it is present in the medium the *lac* operon should be switched off irrespective of the presence of lactose. There is no point expending energy in producing the enzymes necessary for metabolising lactose when glucose metabolism is more efficient. Hence glucose inhibits the metabolism of lactose; in fact it inhibits the metabolism of all other sugars as mentioned above. When glucose is absent however, the metabolism of the other sugars should be enabled, this is achieved using a transcription factor known as Catabolite Activator Protein (CAP) which as the name indicates, acts as an activator. CAP needs the effector molecule cyclic AMP or cAMP to enable it to bind to its binding site which overlaps the promoter of the operon (the structural genes). This enhances the transcription of the lac genes by RNA polymerase, increasing the transcription rate ten folds. Hence there are four possible situations for the *lac* operon summarised in table 2-2.

In summary there are two approaches to controlling the operon, one is used by lactose (or the inducer in general) and termed negative control, while the other by cAMP and termed positive control. The difference is in the effect of the binding of the effector molecule to the transcription factor and consequently on transcription. In the first case, when the inducer binds to the repressor protein (Lac repressor) it prevents it from binding to the operator region and hence allows transcription to start. On the other hand, when cAMP binds to the activator protein (CAP) it enhances transcription. It is clear from table 2-2 that in the case of the *lac* operon, when both controls are acting on the operon, repression overcomes activation.

Table 2-2: The different nutrient conditions and their effects on the *lac* operon.

Glucose concentration	cAMP production	CAP (Bound to DNA)	Lactose concentration	Lac repressor (Bound to DNA)	Operon state	Explanation
Low	High	Yes	Low	Yes	OFF	Activation & repression
Low	High	Yes	High	No	ON	Activation & no repression
High	Low	No	Low	Yes	OFF	No activation & repression
High	Low	No	High	No	OFF	No activation & no repression

The production of cAMP is coupled to the presence of glucose, when the concentration of glucose is high, production of cAMP is low, hence the transcription factor CAP will not be activated and hence the metabolism of all sugars other than glucose will be repressed. In such a scenario glucose acts as a global regulator for all sugar metabolism via cAMP and CAP. The opposite happens when the concentration of glucose is low.

It should be noted that the above presentation of the structure and functional operation of the *lac* operon is highly simplified. For example, structurally the operator region O to which the repressor protein binds is not contiguous, but is dispersed into three different locations that are interspersed with the structural genes and their promoter (figure 2-6).

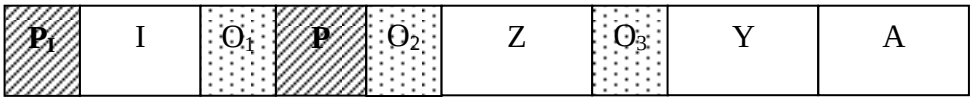


Figure 2-6: A more detailed view of the structure of the *lac* operon.

These structural details have an effect on the functioning of the operon and its expression levels. As explained in table 2-2 above, there are three different cases in which the operon is switched off, however the expression rates in all three, whilst still very low compared to the on case, are not equal. This is because the binding of CAP to the DNA alters its conformation making the binding of the repressor different than when CAP is not present, and both cases are different from the case when neither protein is active (corresponding to the case when both nutrients are present). For a more in depth discussion of the *lac* operon with more details of the regulatory mechanisms involved, see for example (Ptashne and Gann 2002; Santillan and Mackey 2004a), and for a discussion of the metabolism of other sugars and their regulation see for example (Kaplan *et al.* 2008; Kremling *et al.* 2009).

2.8 Other levels of control

In the previous section we have considered the control of transcription, but what if a condition occurs after a gene has already been transcribed, that necessitates regulating it either by stopping its expression completely or changing its expression level? The cell should have the ability to control this level at later stages after transcription, especially in the case of an extreme condition. This is usually done by degrading the molecule to be controlled such as mRNA or even the final product of expression which is the protein. We briefly outline those controls below.

2.8.1 Post-transcriptional control

This refers to controls that are exercised after transcription has taken place but before translation and hence are performed on the mRNA molecule. For Prokaryotes, translation takes place in the cytoplasm which is the free space in the cell. As far as gene expression is concerned, regulation at this stage mainly involves preventing the mRNA from being translated, for example by degrading it.

2.8.2 Control of translation

Like transcription, translation also takes place in three stages, initiation, elongation and termination, where elongation here is of the polypeptide chain formed. Each of these stages is regulated by certain proteins. If translation is allowed to start it can still be controlled while in progress using those proteins, either to halt translation temporarily and resume it later, or to completely terminate it without producing a protein. In the latter case, the resulting polypeptide will eventually degrade.

2.8.3 Post-translational control

Translation of the mRNA into a protein is not the end of the story as this protein needs to be folded in the appropriate conformation and then undergo other chemical modifications such as glycosylation before it is ready to perform its intended function. Often several such proteins are assembled together to form a larger protein, common examples of which are homodimers which consist of two proteins of the same type, and heterodimers involving two proteins of different types. There can also be

assemblies of more than two proteins. The protein also has to be transported to the location in the cell where it will perform its function, a process known as protein targeting. Only then gene expression is complete.

Given that the protein has already been produced, post-translational control of gene expression involves disrupting any of the above processing steps of the protein (e.g. chemical modification or assembly). However, the most important means of post-translational control is by degrading the protein. Degradation can be left to take place naturally which will take a long time, or it can be performed by enzymes that cleave proteins, achieving a much faster result. This process is known as protein lysis, or proteolysis and the enzymes that lyse the protein are known as proteases.

2.8.4 Further levels of control

The different stages of regulation of gene expression outlined above culminate in the protein being ready to perform its functions, i.e. it is available if needed. There are other processes in the cell that may then activate or de-activate the protein in response to different stimuli. Those are normally reversible processes, unlike proteolysis. Examples of such processes include the binding of an effector molecule to the protein as discussed above; a common process among those is phosphorylation which involves the binding of phosphate to the protein and its reverse process of dephosphorylation.

The above discussion relates to producing a protein and activating or de-activating it (effectively switching it on or off), however, if the protein acts as an enzyme, i.e. a catalyst (as opposed to a controlling factor) it may have additional controls. For example, its activity can be modulated in a kinetic manner through different inhibitions by its substrate or product (Fell 1997).

It is obvious that the closer the control action is to the final product of gene expression, the faster the response will be. If one switches a gene off through transcription control then it will take some time for the actual protein levels that this gene codes for to vanish. This is because the mRNA molecules that have already been transcribed before the switch off will still be present in the cell, and they will be

translated into more proteins. Only when those proteins and the ones that already existed in the cell have all been degraded that the complete switch off will be effected. Preventing translation on the other hand will have a faster response as only the proteins that have already been translated will be in the cell as no further translation will be allowed. An even faster response can be achieved by destroying the protein on the spot through proteolysis. On the other hand, if only temporary halting of the function is required, as for example in response to relatively small changes in conditions, then phosphorylation and dephosphorylation result in a faster and more efficient response.

Typical time duration of some of these processes in *E. coli* is as follows: binding of an effector molecule to a transcription factor takes a few milliseconds, transcription of a gene takes about a minute and translation twice as much. The lifetime of an mRNA molecule is a few minutes and of a protein about a couple of hours (Alon 2007a).

This gradation in speed of response has analogous situations in industrial control where the closer the control action is to the process variable the faster is the response. Such concepts are utilised in cascade control where a control loop is nested inside another. The inner loop has faster dynamics than the outer one and affects the control action much more quickly, a strategy often employed in distillation columns control (Stephanopoulos 1984).

Speed of response whether in a cell or an industrial process comes at a price, i.e. there will always be a trade off. Damaging an mRNA molecule or a protein after going through the long process of transcription and/or translation means that large amounts of energy have been wasted. However, this is imperative if the fast response is required for the very survival of the cell. An analogy in an engineering system would be hitting the brakes of a car to prevent it from crashing. The energy of the car is wasted as heat energy in the tyres and may even damage the tyres, but this is imperative to save the whole car or prevent injury to its passengers.

2.9 "Omics"

The genome of an organism contains all its genes in the DNA, whether expressed or not. Some of those genes will be transcribed under the appropriate conditions, while others may never be transcribed. The set of all possible transcripts, i.e. mRNA molecules generated by the transcription process is referred to as the Transcriptome; again not all of those will necessarily be translated into proteins. The set of all proteins that can be produced from the mRNA of the organism, irrespective of whether they are actually produced or not is known as its Proteome. Some of the proteins will act as transcription factors through interacting with other proteins, effectively involved in on/off (logic) control, those form the Interactome. Other proteins will act as catalysts in the metabolic processes of the organism and will contribute to the Metabolome, which is the set of all chemicals that are processed inside the cell. Those different layers of cellular functionality and the enormous amount of data they produce are collectively referred to as "omics" (figure 2-7). A wide array of advanced analytical techniques is used to generate this omics data through carefully designed experiments, for an overview of such techniques see (Lorkowski and Cullen 2003; Lay *et al.* 2006).

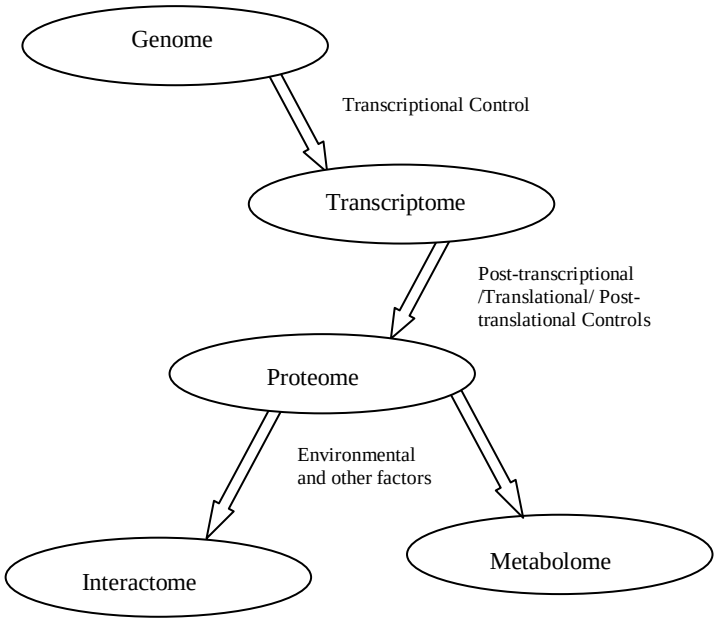


Figure 2-7: Omics and their relationships – feedback paths between the different layers not shown.

The different layers of functionality as depicted in figure 2-4 and the corresponding figure 2-7, interact with each other to yield the behaviour exhibited by the cell. For example, transcription regulation determines which part of the genome will go into the transcriptome, i.e. which genes will be transcribed. From a mathematical standpoint, this can be regarded as a mapping from the genome to the transcriptome under the transcription regulation function. Similarly, post transcriptional, translational and post translational controls specify another mapping, that from the transcriptome to the proteome. The conditions in the cell environment, for example the concentrations of the different nutrients and other molecules required for metabolism, will determine the activity of the different enzymes (part of the proteome). This is manifested in the fluxes within the metabolic network (the metabolome), and can be viewed as a mapping from the proteome to the metabolome. Functionality of another part of the proteome, that which is concerned with regulatory proteins is mapped to the interactome. In a mathematical sense, one can think of the resultant effect of the interactions at the different layers as a composition of these mappings. It should be noted that these different functions are affected by both external and internal conditions to the cell. Furthermore, the interactions involve feedback both within a layer as in the case of enzyme activity, and between layers as in the case of the regulation of gene expression.

The overwhelmingly large amounts of data produced by the omics experiments pose a challenge in their analysis and interpretation, requiring mathematical and computational tools to undertake this task (Wolkenhauer *et al.* 2003; Lay *et al.* 2006; Mehta *et al.* 2006; Selzer *et al.* 2008). Thus the mathematical view of regulatory functions outlined above coupled with the engineering view alluded to several times earlier, provide a powerful basis for assimilating and understanding this data. The approach often used by engineers in attacking such problems is mathematical modelling. There are several modelling approaches employed and we will look at the most commonly used ones in the next chapter.

2.10 Summary and Conclusion

The purpose of this chapter was to introduce the regulation of gene expression to engineers as a prelude to developing a mathematical modelling method later in this

report. The strategy followed to achieve this was two pronged; firstly to simplify the treatment as much as possible, and secondly to use analogies with engineering systems.

There are two outcomes from this chapter relating to this strategy, the first is to give an overview of the regulation of gene expression, with particular emphasis on Prokaryotes (bacteria). To reach this outcome, we first had to provide the foundation to build upon, which involved presenting some basic concepts from molecular biology. This equipped us with enough knowledge to answer three main questions important for the understanding of the regulation of gene expression, namely what is gene expression, why does the cell need to regulate it and how does it do that?

It is the last question that we elaborated upon most, indicating the different levels of gene regulation, namely transcription, post-transcriptional, translation and post-translational regulations. We have placed special emphasis on transcription regulation because it is the most studied and the best understood. It also makes more sense for the cell to control gene expression at its inception rather than at a later stage, hence avoiding wasting the energy spent getting to that stage. We also covered the *lac* operon in the bacterium *E. coli* as an example of transcription regulation. This helped us elucidate some of the fundamental concepts in transcription regulation, such as gene activation and repression, constitutive and regulated genes and basal expression level; and also the main players in this process such as a transcription factor, promoter and effector molecule. In covering the above, Prokaryotes were chosen rather than Eukaryotes because of their wide use in industry, their simpler composition and because they are well studied. It should also be pointed out that the treatment was highly simplified, especially with regards to the details of the molecular mechanisms such as the binding of RNA polymerase to the DNA and the role of transcription factors on a molecular interaction level.

Discussion of the regulation of gene expression at the different levels of cell functionality led us to a discussion of the different omics including genomics, trancriptomics, proteomics and metabolomics, and the proliferation of the corresponding types of data. Such a large amount of data is impossible to make sense of intuitively; mathematical and computational tools are needed to assimilate all this

data in a meaningful way. Hence this makes a case for the use of mathematical modelling to study the regulation of gene expression.

The second outcome from this chapter is to point out throughout the presentation and wherever appropriate, the similarity between regulation of gene expression in a cell and regulation of an industrial process. Consequently this suggests that the means used for the analysis of industrial processes in particular mathematical modelling, can be used in studying gene expression, further reinforcing the case for the use of mathematics in such an endeavour. So how can we apply mathematics to the modelling of the regulation of gene expression? This is the topic of the next chapter.

Chapter 3: Modelling the Regulation of Gene Expression

3.1 Introduction

The purpose of this chapter is to examine how to build mathematical models of the regulation of gene expression and to briefly survey some of the more common modelling methods used. Before doing that however, we need to outline some key concepts related to modelling. In the previous chapter we have described gene expression and how it is regulated, we also made the case for the use of mathematical modelling to understand this regulation. In this sense, mathematical modelling is our research method, and as with any method, before employing it one has to be aware of its limitations and potential pitfalls. Hence we start this chapter with a discussion of some of the theoretical issues related to modelling and the underlying concepts. In order to build models of real life systems, a modeller needs to make some choices and decisions; hence after discussing the theoretical issues we will need to address some of the practical issues involved in modelling. Among the decisions a modeller has to make is the level of abstraction at which he will consider the phenomena being modelled and how much detail he is willing, and able to incorporate into the model. Armed with this knowledge, we will then survey some of the common approaches to modelling the regulation of gene expression. Rather than derive mathematical formulations, we will look at the big picture and classify the models according to different criteria, both biological and mathematical. There is a wide choice of mathematical formalisms available to the modeller, and amongst those by and large the most common is modelling using differential equations which gives quantitative models. Hence we will demonstrate how to apply them in a generic way, to the modelling of gene expression regulation presenting their advantages and also pointing out some of their shortcomings and of quantitative methods in general. This will lead us to consider models of a qualitative nature, among which, one of the most widely used are Boolean models. Those too have their shortcomings which we will discuss, paving the way to proposing our method.

3.2 Mathematical modelling concepts and caveats

The purpose of modelling a given system is to understand how it functions in order to be able to predict its behaviour and possibly to ultimately control it. In this section we briefly touch on some of the theoretical issues underlying the process of modelling, those can be described as meta-modelling issues. The purpose is to point out to some of the conceptual limitations of modelling, and hence to set realistic expectations with regards to the results the models provide. This reality check is important because of the growing role of modelling in modern biology as evident in the proliferation of the emerging discipline of Systems Biology, currently a very active area of research.

A model of a system is essentially a representation of our perception of the system rather than of the system itself (Casti 1989). This applies to any modelling approach but here we are primarily interested in mathematical models whereby systems are described by equations and where numerical values may be assigned to some system parameters.

We first denote briefly what is meant by a system and the state of a system. The IEEE Standard Dictionary of Electrical and Electronic Terms defines a system as "*a combination of components that act together to perform a function not possible with any of the individual parts*" (Radatz 1997), such functionality of a system that is not present in its components but results from the interaction of those components is termed "emergent" functionality (Nagel 1961). Whilst intended for engineering or physical systems, the above definition is general enough to encompass other forms of "systems" such as biological, economic or even social systems. Hence, a system is not necessarily tangible, but it does need to be "observable", meaning that one should be able to make observations about it, whether qualitative or quantitative. The observations describe the state of the system, which is the second notion we want to discuss. A state is one of those concepts that are usually understood intuitively but are hard to articulate in a formal definition. Nonetheless, a state is taken to indicate the information about the system at a given time instant that is sufficient to completely describe the system at that instant (Cassandras 1993). A familiar example for engineers is in thermodynamics where the state of a system is described by various state variables such as pressure, volume, temperature and entropy. It should be noted

however, that a state need not have a physical interpretation as it is an abstract notion (Casti 1989).

The above discussion then implies that the observation mechanism should be able to distinguish between different states, if it cannot then the states are considered equivalent with respect to this particular observation mechanism, even if they are different in reality. This then prompts the question, what is reality and how do we know whether the observations do or do not represent it? This raises philosophical questions related to acquiring knowledge and representing it, i.e. epistemological and ontological issues (Nagel 1961). In summary, we can conjure up a mental image as depicted in the Venn diagram in figure 3-1, whereby we have some system whose behaviour is described by states. We can then say that reality represents the set of all possible states of the system (the universal set U in the diagram), some of which will be observable (the set S) and among those, some will be distinguishable from each other (the set X). From that last set of states we can use a subset to build a model. All those sets are in an inclusion relationship in the set theoretic sense, i.e. each set includes the one following it in the above description, as depicted in figure 3-1.

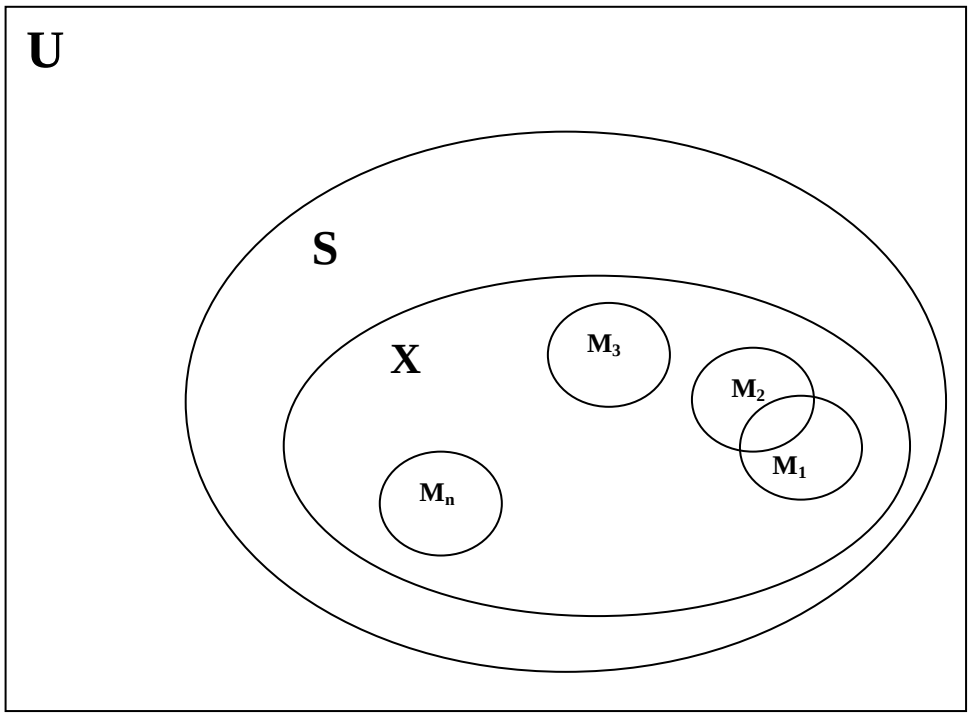


Figure 3-1: Venn diagram depicting sets of states of a given system and their relationships.

Different subsets of the overall set of observations represent different aspects of the system behaviour and can be used to build different models (the sets M_1, M_2, .., M_n). Note that two (or more) of the observation subsets used for building a model may intersect, which means that some aspect of the system behaviour can be described by more than one model. On the other hand two such subsets may be mutually exclusive indicating that there is no model that can capture both aspects of the system behaviour. One such example is in quantum mechanics with the dual particle/wave aspect of elementary particles where only one aspect of their behaviour can be observed under a given experimental setup.

Of course figure 3-1 can provoke further philosophical questions, for example how do we know the boundaries of the universal set U if it is not completely observable? In fact how do we know whether anything other than the observable actually exists? As we have mentioned, this is a mental image and is based on past human experience. For example, according to the theory of evolution bacteria existed long before humans did, yet we only became aware of their presence in the seventeenth century after the invention of the optical microscope (Porter 1976). Similar arguments hold for many other areas of human endeavour. This means that the set of observable states grows with the advancement of technology. As with the example of the microscope, many other technologies allow us to know things now that we did not know in the past such as the different omics information discussed earlier. Undoubtedly new technologies will be developed with time that will expand the observable set S further within the universal set U. Furthermore, the set of discernable states X among those observable, expands with the advancement of technology as well. Consider temperature measurement for example, for a thermometer with resolution of one tenth of a degree, the two temperatures 25.42^0C and 25.43^0C are the same, i.e. they are equivalent states with respect to this particular thermometer (or observation mechanism in general). With a higher resolution thermometer they become two distinct states, hence the set X becomes a larger subset of the set S. Will S or indeed X ever reach U? This is an important question that is beyond the scope of this work and falls more in the realm of the philosophy of science, for more details see for example the work of Karl Popper or Thomas Kuhn (Nagel 1961; Casti 1989; Casti and Karlqvist 1990). We will thus cease this line of thought at this point and resume our discussion of modelling.

Observations tell us what happens but not how it does. Because a collection of data does not in itself constitute knowledge (Duncan 2007), there is a need for a means to relate the observations in a meaningful relationship. As suggested in the previous chapter, this is a role suitable for a mathematical model which can be viewed as a representation of the observable reality in some formal mathematical system. In a more abstract sense, a model is a mapping from the set of observations to the set of states described by the model (Casti 1989). Ideally those two sets should be the same, meaning that the model should be able to reproduce the set of observations, i.e. describe it as it is (a descriptive model). However, a model is more useful if it can also predict the behaviour of a system in addition to describing it, i.e. a predictive model. The predictions produced by the model are the results of derivations and mathematical manipulations of the model, which are then translated to expected observations. This means that the model should be able to reproduce observations that lie outside the set of observations on which it was built. To account for un-modelled features, modellers often resort to adding stochastic terms to the mathematical description to embody the uncertainty about the knowledge of the system. The uncertainty is assumed to be due to either aspects of the system behaviour un-accounted for in the model, or noise (error) in the observations accounted for (Kazakos and Papantoni-Kazakos 1990). Inevitably there will be discrepancies between the data produced by the model and those recorded from experiments, functions of such discrepancies (usually statistical) can be used to judge the quality of the model. It should be pointed out that even a non-predictive model or theory in general (i.e. not necessarily mathematical) can still have great explanatory power and hence be very useful. A highly celebrated example of such a case in biology is the theory of evolution which describes the evolution of the characteristics of a species but it does not predict how it will change in the future. In other words, knowing the environmental conditions we cannot predict the genetic makeup of the emerging species nor even its physiological description. In essence the theory of evolution tells us how we got here, but does not tell us where we are heading, thus it describes an observation but cannot predict its future course (Casti and Karlqvist 1990).

The above discussion places two types of constraints on the accuracy and hence usefulness of a mathematical model, in particular its predictive power. The first type of constraints relates to the set of observations because as outlined above, our

knowledge of the system is limited by what we can observe of its behaviour. Hence, whilst the model is based on a subset of the total observations, it attempts to make prediction about aspects of the system behaviour that may not be reflected in these observations, potentially undermining the accuracy of the model. The second sort of constraints is the mathematics being used, as different modelling formalisms are better suited to different investigations of a system's behaviour, and can support different mathematical derivations, leading to different results and predictions some of which may be more accurate than others.

Because we can use different subsets of observations in formulating a model (figure 3-1), there can be more than one model of the same system each describing some aspect of its observable behaviour using potentially different mathematical formalisms. We will discuss some of these formalisms below and in later chapters. Furthermore, once we have chosen a particular subset of observations, we can still have more than one model describing the same set, providing different views and different inferences. Those are considered equivalent models that are related to each other by some form of "transformation", an example familiar to engineers is time domain and frequency domain descriptions of a system, related by the Fourier transform. This issue will also be discussed in later chapters.

To give a concrete example of these abstract notions we consider a liquid storage tank. We may be interested in the level of the liquid in the tank which is a continuous variable taking real values lying between zero and some maximum corresponding to the height of the tank. This case can be modelled by a simple differential equation relating the rate of change of the level to the inlet and outlet flow rates and parameterised by the tank cross sectional area. Such a model is often used in regulatory control of the liquid level. Alternatively, we may only be interested in whether the liquid level exceeds a certain point in the tank above which there is a possibility of spillage and hence a potentially hazardous situation, especially if the liquid is flammable or toxic. From such a viewpoint the liquid level can be in one of two states, either above or below the hazardous point, a situation that can be conveniently described using Boolean algebra. Such a model is used for the design of a safety shutdown system that may override the regulatory control of the level.

This simple example highlights the ideas discussed above in that different observations may require different mathematical tools to develop different models for the same physical system. It further illustrates that some of the models may be quantitative in nature such as the one modelling the actual liquid level in the tank whilst others may be qualitative such as the one describing whether the level is above or below a certain value, irrespective of how far it is from that value.

Casti (1989) summarises these ideas in stating that "*a model is a mathematical representation of the modeller's reality, a way of capturing some aspects of a given reality within the framework of a mathematical apparatus that provides us with a means for exploring the properties of that reality mirrored in the model.*" Note that as mentioned at the end of this quote, only the properties of the system "*mirrored in the model*" can be studied by it, emphasising that investigating different aspects of the system behaviour may require different models.

From an application stand point, formulating a mathematical model, i.e. describing the (observable) real life system in mathematical terms requires knowledge of the system at hand, i.e. domain specific knowledge. Once the system is described mathematically however, it becomes a mathematical problem and a battery of methods is available for its investigation and manipulation including analysis, synthesis and optimisation methods. The results obtained have then to be interpreted from a domain specific viewpoint for a sanity check as some results of the analysis while mathematically sound, may be physically meaningless such as obtaining negative values for parameters (figure 3-2). In this sense mathematics can be thought of as a language and modelling as a translation from one language, the domain specific knowledge into another, the abstract mathematical knowledge. Indeed, modelling is essentially a process of abstraction that divorces the system from its domain specific setting and transforms it into a mathematical entity. When doing so we find that oftentimes systems that are distinct in real life are modelled by the same mathematical description. For example the second order linear differential equation with constant coefficients describes both a mechanical system of mass, spring and damper, and also an electrical system of inductance, capacitance and resistance. The abstractions of those physical elements are the notions of inertia which resists motion, stiffness which stores energy and dissipation which dissipates energy (Doebelin 1980;

Cha *et al.* 2000). This proves further that a model is just a mental construct with no intrinsic physical significance, only when related to a particular physical system does it acquire this significance.

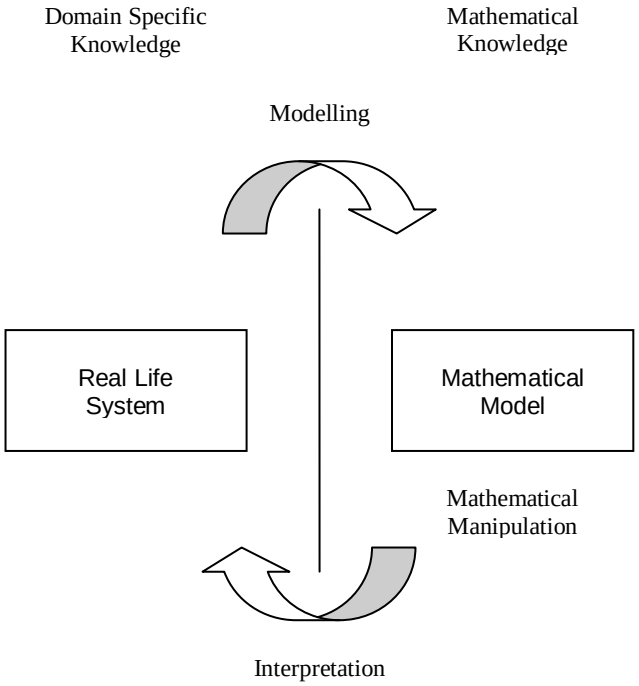


Figure 3-2: Modelling as a process of abstraction.

To summarise the main points raised in this section we state the following

- A model describes a subset of the observable reality
- There can be more than one model for a system, each corresponding to a particular subset of observations.
- For a given subset, there can be equivalent models where equivalence is meant in the sense that they describe the same set of observations but with different mathematical machinery (usually related by transforms).
- Modelling is an abstraction process which means that different systems when abstracted from their implementation details can end up with the same mathematical model.
- The predictive power of a model is limited by both the observations on which it is built, and the mathematical formalism used to build it, including any assumptions related to both.

The essence of this section is captured by a quote attributed to the British statistician George Box in which he says "*all models are wrong, some are useful*", a rather cynical variation of which is given by Wolkenhauer and Ullah (2007) as "*all models are wrong, some more than others*".

3.3 Model building decisions

This section addresses issues that mirror the theoretical ones discussed above, representing their applied counterparts. Here we consider the decisions a modeller needs to make when embarking on a modelling task. For our purpose we will consider the model of a system to be a representation of some aspects of the system that are of interest to the modeller. Two key concepts are embedded in this statement; the first relates to the phrase "some aspects" and the second to the phrase "of interest to the modeller". Those two concepts correspond to similar ones in the theoretical discussion above relating to the subset of the set of observations to use and that the model investigates aspects of system behaviour that are reflected in the model.

The notion that a model represents "some aspects" of the system implies that the act of modelling involves a simplification of the system behaviour. Thus it is not only acceptable that a model ignores some aspects of that behaviour, but in fact it is expected to do so. It would be impractical to expect a model to represent every feature of the system, as in such a case it ceases being a model and becomes a replica of the system. From a practical standpoint then, an important decision in the modelling process is, which aspects of the system behaviour to ignore and which to include in the model. The answer to this question leads us to the second concept and that is that the model has to address the issues "of interest to the modeller", i.e. those for which the model is formulated in the first place. Therefore the model should ignore aspects of the system behaviour that are believed not to contribute to or at least not to strongly influence the function being studied.

To illustrate these concepts let us look at a concrete example. Consider a metal rod exposed to heat, this system can be studied from different engineering perspectives. A mechanical engineer could be interested in the expansion of the rod. A materials engineer may be interested in the effect of heat on its tensile strength, a metallurgist

in the molecular structure, an electrical engineer in the effect of heat on its resistance, a communication engineer on its electromagnetic properties in case it is being used as an antenna. Different equations and hence models relate the different properties mentioned above to heat, those range from straight forward linear algebraic equations in the case of linear expansion or change in electrical resistance to vector partial differential equations in the case of heat transfer and electromagnetism. To attempt to formulate a single model that captures all these behaviours of the system is a futile endeavour, simply because the "interest" of each of those engineers is different. Hence the scope of the model will have to be limited to the features and functions of the system relevant to this interest.

Within the scope of the model, the modeller needs to decide on the level of abstraction at which the system being modelled will be viewed. Normally the more abstract the view is the less detail about the system will be needed, as for example with the case of the liquid level in a storage tank mentioned above. Sometimes however, it is the nature of the details that changes rather than the amount. For example when studying a chemical reaction in a stirred reactor, will the modeller investigate the behaviour of the bulk liquid and how the reaction will be affected by the mixing speed, or will he study the kinetics of the reaction irrespective of the reactor, or possibly only consider a stoichiometric approach? Each of these levels of abstraction involves different types of details about the reactants and the vessel. Hence, once decided on the level of abstraction, the modeller has to further decide on the amount of detail to include in the model.

For example, when considering the bulk liquid we can ask whether it will be considered homogeneous or not, if not how will its composition change with location in the vessel? Another issue that comes up in some situations is directionality, i.e. is some property say viscosity the same in all directions (anisotropic) or does it have different values in different directions possibly because of lack of homogeneity of the liquid due to inadequate mixing? Furthermore, are the system parameters - such as properties of the liquid or the vessel - constant or do they vary with time, and if they do, is this variation deterministic or stochastic (random)? Will those parameters be treated as lumped or distributed? For example in electrical engineering the resistance of a wire is effectively distributed over its length, however, it is often treated as

lumped. This treatment may be valid under certain circumstances and invalid under others, for example when the wavelength of the current in a wire is of the same order of magnitude as the length of the wire, the wire will start to act as an antenna and can no longer be treated as a lumped element (Doebelin 1980). These concepts may also apply to the variables being investigated and not just the parameters, as for example with the case of fluid particles in a pipe where the velocity will depend on their distance from the pipe wall.

The exposition above indicates that, depending on the amount of detail included in the model, it can become very complicated. Hence the modeller should include only the details that serve the purpose of the model, in a sense using Occam's razor, i.e. that one should use the simplest model possible that adequately describes the system behaviour (Gershenfeld 1999). Whilst this argument calls for simplifying the model, one should be careful not to oversimplify as this may give misleading results. Hence, there will always be a trade-off between accuracy attained by including more details in the model and simplicity attained by ignoring some details. The modeller has to strike a balance between those two objectives. Indeed, when discussing modelling, Gershenfeld (1999) indicates that "*Many efforts fail because of an unintentional attempt to describe either too much or too little*".

Having decided on what to model and the amount of details involved, next the modeller has to decide on which modelling method to use. This will not only depend on what is being modelled but rather paradoxically on the modelling method itself, where sometimes a method is used solely for the sake of mathematical tractability. Fitting a system in a given mathematical framework may require many simplifying assumptions regarding its behaviour. For example a system may be assumed to be linear, primarily to enable benefiting from the wealth of methods available for linear systems analysis, as non-linear systems are difficult to analyse. Indeed, Naylor and Sell (1982) capture this further trade-off when stating "*The formulation then of a mathematical model is a compromise between mathematical intractability and inadequate description of the system being modelled*.", echoing the earlier quote by Gershenfeld albeit from a different perspective, that of mathematical tractability rather than of amount of detail.

The paradox referred to above comes from the fact that on the one hand choosing a method depends on what the modeller wants to investigate, and on the other the method chosen constrains what the modeller can investigate. This emphasises again the notion of the mathematical formalism constraining the usefulness of a model as outlined in the theoretical discussion above (Casti 1989).

There are several mathematical methods that can be used in modelling, we will only highlight the main ones, but perhaps more importantly we will classify them. One classification is into dynamic versus static models. Models of dynamic behaviour describe the change of a given variable normally with respect to time in response to change in another one or more variables. Hence they are useful in studying such features as speed of response and the related dynamic characteristics such as damping and delay. Static models on the other hand describe a relationship between the variables that conveys the dependence between them without regard to time. For example a stoichiometric equation is a static relationship between the reactants and how the products depend on them, but it does not contain any information about timing and hence the speed of the reaction. On the other hand, kinetic models contain rate and hence timing information and can thus be used to determine quantitative information about the concentrations of the reactants and products at different points in time and how fast they reach those concentrations. Another example is in networks whether road networks, communication networks, piping networks or more relevant to our work here, gene regulatory networks. Static information is essentially embodied in the network topology, i.e. connectivity, for example the number of routes that connect two cities whether directly or indirectly through intermediate cities. Dynamic information on the other hand is contained in the traffic patterns on the roads between those two cities such as issues of congestion and throughput, which will determine how fast it takes to go through each route, i.e. issues of timing. A third example is in process control where static information is used in the shutdown system (logic control), as for example in the tank liquid level scenario discussed previously stipulating that if the level in the tank exceeds a certain point then shutdown this particular unit. Dynamic information on the other hand would be used in regulatory control where it would help specify the controller gains and timing (integral and derivative) to be able to control the speed of response of the change in liquid level and how far it can deviate from the desired value. It is clear that whilst dynamic

models provide more information about the system being modelled in particular quantitative information, they in return require more information for building a model, again quantitative one. This causes two concerns; firstly that often such information is very difficult to obtain, such as kinetic parameters for a reaction. Secondly the accuracy of the results obtained from the model will be limited by the accuracy of this information, in addition of course to the accuracy of the model itself as discussed above. Those concerns undermine the advantages of dynamical models or any models providing quantitative information in general.

Dynamic behaviour can be modelled using continuous time in which case they are formulated using differential equations. Alternatively, values of the variables of interest may only be available at discrete points in time, as in the case of sampled systems, hence leading to difference equations. The values of the variables being investigated can also be assumed to be continuous or discrete, the latter leading to a discrete event systems formalism such as finite automata and Petri nets. Another classification is into deterministic versus stochastic models, where the latter means that the variables are assumed to be random processes, hence characterised by probability distributions. A deterministic description on the other hand does not contain this probabilistic aspect. The three classifications mentioned above namely static v dynamic, continuous v discrete and deterministic v stochastic are orthogonal in the sense that a model can belong to either type of each of the three classifications, e.g. dynamic discrete deterministic or dynamic continuous stochastic, etc.

We can summarise this section in two main points. The first concerns modelling decisions and the second concerns modelling errors, reflecting the choices and caveats involved in modelling.

Modelling decisions

There are three main decisions the modeller needs to make, as depicted in figure 3-3, usually in the following order

- The level of abstraction at which to view the system
- How much detail to include in the model
- The mathematical formalism to use

Each of these decisions can have sub-decisions as outlined above.

Modelling errors

There are two main conceptual sources of error that are a result of simplifications in modelling, those are;

- Simplification due to ignoring some aspects of the system behaviour
- Simplification for the sake of mathematical tractability

Those two sources of error are different from errors due to measurement. Modelling errors are deliberate in the sense that the modeller consciously chooses to make those simplifications, while measurement errors are inevitable as they are ultimately governed by the technology available.

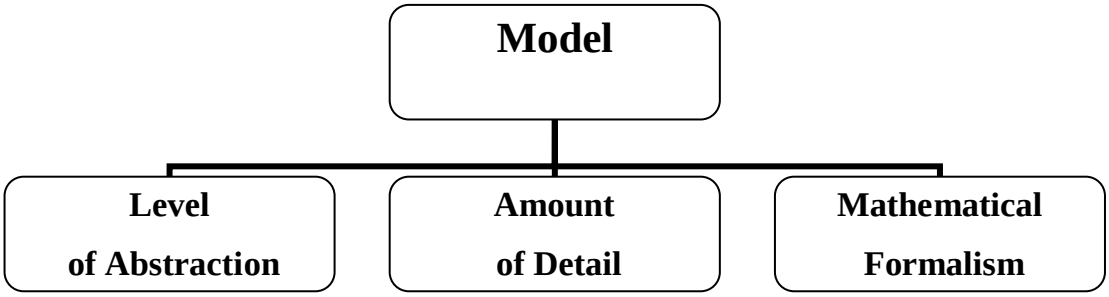


Figure 3-3: Modelling decisions.

3.4 Mathematical modelling in biology – Systems Biology

Mathematical modelling has been utilised in biology for a long time, at least since the early twentieth century in the work of Lotka and Volterra in the 1920's on modelling population dynamics involving prey-predator relationships (Rosen 1970). The following decade, the journal "Bulletin of Mathematical Biology" was launched in 1939.

As the knowledge of biological processes grew, especially at the molecular level following the work of Monod and others in the 1960s, there was a surge of interest in applying mathematics in biology in particular using a dynamical systems approach. Several books appeared at the time that formally applied dynamical systems theory to biology, for example the book by Rosen in 1970. This was also evident in the launch of several journals devoted to the subject around that time such as Journal of Theoretical biology in 1961, Mathematical Biosciences in 1967 and Journal of

Mathematical Biology in 1974 among others. With the recent proliferation in the quantity and quality of omics data as outlined in the previous chapter, a new wave of interest in applying mathematics to biology emerged. What sets this new wave apart from earlier ones is that the scope (several sub-cellular processes), the data (vast amounts of high quality data) and the mathematical tools utilised have expanded greatly (Kitano 2002; Wolkenhauer 2007). But perhaps more important than the mathematics is the shift in the underlying conceptual view to modelling, evident in the new emphasis on the interconnectivity of systems rather than on an individual system, which resulted in some philosophical discussions on "holism" versus reductionism (Kell and Oliver 2004; Van Regenmortel 2004; Cornish-Bowden and Cardenas 2005; Noble 2008; Gatherer 2010). This holistic view has led to the birth (or some would argue reincarnation) of the new interdisciplinary field of Systems Biology (Cornish-Bowden 2005; Noble 2008). Like the previous waves of interest in applying mathematics to biology, this one has also led to the publication of several books and the launching of several journals, for example Systems Biology in 2004 (later IET Systems Biology), Molecular Systems Biology in 2005 and BMC Systems Biology in 2007. From such a perspective, Systems Biology is yet another phase in applying mathematics in biological research, with its own scope and tools that match the current research questions in biology and reflect the current technology, i.e. the availability of omics data.

The above is not meant as an account of the history of mathematics in biology, but rather pointing out some of the milestones in this history related to adopting a Systems approach. Unsurprisingly there are different views regarding the usefulness of such approaches in molecular biology (Cornish-Bowden 2005). Some are so sceptical of Systems biology as to state that "*Because it is so broad and has few recognized boundaries and plenty of funding, it is attractive to anyone who has ever thought about life and has some relevant technical expertise.*" (Werner 2007). Others on the other hand are overly optimistic and view Systems Biology almost as a panacea that will usher in a new era of biology, and describe it as a "paradigm shift" that will cause us to re-examine the philosophical basis of biology and will eventually lead to answering the question "what is life?" (Westerhoff *et al.* 2009). Our view however, is a more pragmatic one; we believe that modelling in its current form is just another tool added recently to the biologist's toolbox borne out of need. Note that

here we are talking about biological systems in the sense of cell biology rather than population biology and epidemiology which have used mathematical models for decades as mentioned above. In cell biology the emphasis of research has changed over the years, with each phase necessitating new tools. In the past it was mostly about biochemistry and hence the tools used were concerned with chemical composition, reaction characteristics and physical properties. Those have led to the discovery of the double helix which ushered in the era of genetics and molecular biology in general; one strand of this led with time to the change of emphasis from metabolites to the genes that code the enzymes catalysing the metabolic reactions. This necessitated the development of tools that can deal with genes and DNA such as PCR and related techniques. Again research in genetics led to the study of gene expression at its different levels, transcription, translation and post translational modifications of proteins, which led to the development of all the "omics" tools, as discussed in the previous chapter. Those tools generated large amounts of data, but because more information does not necessarily mean better understanding, there was a need for a means to assimilate all this data, investigate the different levels of functionality and how they interact, and interpret the results. Mathematics came as a fitting candidate for this job. Of course along all those developments there was a development in bioinformatics that matured greatly with the proliferation of the "omics" disciplines. Hence in such a context mathematics, in particular from the perspective employed in Systems Biology, can be viewed as a tool needed to address the recent problems that have arisen in biological research. Again in a sense it is merely a phase in the natural progression of biological research; or to use biological terms, it was "naturally selected" because it was the "fittest" tool for this particular period in the "evolution" of biology.

3.5 Modelling the regulation of gene expression

In this section we survey the main mathematical approaches used in modelling the regulation of gene expression and their conceptual basis. There have been several excellent surveys in the literature of this topic over the past few years, for example (Smolen *et al.* 2000; Wessels *et al.* 2001; De Jong 2002; Ideker and Lauffenburger 2003; Schlitt and Brazma 2007; Hecker *et al.* 2009). Hence, rather than simply transcribe such sources here, we will instead take a more abstract view whereby we

will classify the main modelling approaches and for each we will discuss the underlying assumptions, the advantages and limitations of the resulting models and their use. This, we believe, is more instructive than the actual mathematical formulation and its details which can be pursued in any of the pertinent references cited. In presenting the different modelling approaches we will apply the concepts discussed above, namely the level of abstraction, the amount of detail and the mathematical formalism used. Naturally there are several ways to classify such models; we will consider three such classifications, based on scale, function and mathematical formalism.

The levels of abstraction adopted in many modelling approaches are based on scale, i.e. the number of genes studied and included in the model. We remind ourselves that we want to model the regulation of gene expressions, i.e. under what conditions will the different genes be expressed and to what level of expression. Those conditions - normally conveyed by effector molecules - are mediated to the genes through transcription factors, which are in turn products of other genes. Thus the highest level of abstraction, i.e. the largest scale, is the one that would include all the regulated and regulating genes in the organism's genome, sometimes referred to as the regulatory genome (Davidson 2006). Some of those genes will be producing transcription factors in response to external and internal signals as discussed in chapter two, while others will be the target of those transcription factors. Oftentimes genes would be controlling and being controlled by other genes in positive and negative feedback loops, and resulting in an interconnection network of genes. Some genes also regulate their own expression in a form of auto-regulation (Alon 2007a).

As an example, the gene regulatory network for yeast contains more than two thousand genes and more than a hundred transcription factors. Some genes are affected by more than one transcription factors and some transcription factors affect more than one gene in a network of interactions (Lee *et al.* 2002). The association between the target genes and the transcription factors is established through transcriptome experiments whereby the expression levels of the genes are measured and those that are above some threshold are clustered together. A time series of measurements is taken at different points in time and these associations are followed; persisting ones indicate that the genes in a cluster are co-regulated. Bioinformatics

tools and analytical experiments determine which genes produce the transcription factors and what their target genes are. There are some pitfalls to be aware of in this approach, firstly that the expression threshold used for clustering is often subjective in nature and will depend on the researcher and what they are trying to study and their own bias. Secondly, co-expression does not necessarily mean dependence or causation, it may merely mean correlation or association. It should also be noted that transcriptome experiments are performed under specified conditions for the cell, and consequently different conditions may lead to different interaction networks. This is in contrast with the genome which is static in that it does not normally change throughout the lifetime of a given individual of an organism.

So what can be studied at this scale and what mathematical methods can be used? As indicated above, on an abstract level, a gene regulatory network is an interconnection network, other examples of which have been mentioned above. The method commonly used for studying networks is graph theory (Carré 1979). In its most basic form, a graph is described mathematically by two sets, a set of vertices - sometimes referred to as nodes - and a set of edges connecting those vertices in pairs. The nodes can represent any entities the modeller is interested in investigating and the edges represent the relationships between those entities. Engineering examples of graphs representing networks include railways, piping, electrical distribution and computer/communication networks as has been described above. The nodes for those can be train stations, pumping stations, electrical substations and servers/switches respectively. The edges can be the appropriate corresponding connections between those nodes; in the examples above those represent physical connections such as railroads, pipes or wires, however, generally speaking this need not be the case. Connections can also represent information flow rather than material flow.

In terms of details included in models represented by a graph, the most basic information is the network topology, i.e. the connectedness, indicating whether two nodes are connected or not. Additional layers of information can be added on top of that, for example directionality, such as the direction of traffic on a road network or of flow in a piping network, resulting in a directed graph. Furthermore quantitative information can be included such as distance between two points on a road network and are indicated as weights on the edges connecting two nodes.

When applying the concepts from graph theory to modelling gene regulatory networks, we find that nodes can represent genes, and edges represent the regulatory relationships between those genes. Directionality would indicate which gene affects the other, while a sign on the edge (positive or negative) would indicate whether this effect is activating or inhibiting, thus providing further details.

As with other application domains, the graph theory approach allows investigating issues of connectedness which in the context of gene regulatory networks indicate regulatory effects. For example, analysis by graph theory can reveal whether two genes are connected through some regulatory route which may be indirect and hence might not have been detected by past experiments. This can serve to generate hypotheses that can be tested experimentally. Graph theory can also reveal whether there is more than one route connecting two genes hence indicating redundancy that may explain why when some genes are knocked out the cell still carries out the function believed to be coded by those genes.

Another property of networks that can be investigated using graph theory is the in-degree and out-degree of a node, which refers to the number of edges with input arrows to the node and those with output arrows respectively. In the context of gene regulatory networks the in-degree of a gene (node) indicates the number of transcription factors regulating it. Similarly the out-degree indicates the number of genes regulated by the transcription factor produced by this gene. For regulatory genes, the out-degree is often much higher than the in-degree, indicating that such genes control many others, while being controlled by a limited number of factors themselves (Davidson 2006; Alon 2007a). Genes that control a large number of other genes are known as hubs, an example of which is the gene producing the CAP protein utilised in sugar metabolism as discussed in the context of the *lac* operon in the previous chapter.

A recent fairly exhaustive (and exhausting) coverage of the application of graph theory in biology is given by Lesne (2006). A more readable account is given by Alon (2007b), while applications to bacteria and yeast are given by Christensen *et al.* (2007) and Lee *et al.* (2002) respectively. Note that the same graph theoretic concepts are used in modelling metabolic networks and signal transduction networks as well.

Further details can be added to the gene network in the form of probabilistic information leading to Bayesian networks. The idea is that given the dependency of some genes on some other controlling genes, one can establish a directed graph in the form of a decision tree, wherein a parent node would represent the controlling gene and the children nodes represent the genes controlled by it. The probability of a child gene being expressed will clearly depend on the probability of expression of the parent gene leading to conditional probabilities for the different genes in the network (Friedman *et al.* 2000; Needham *et al.* 2006). Note that the terms parent and child in this context refer to their position in the tree and not to a biological progeny relationship. The Bayesian approach is intuitive and provides further value to the graph, but suffers from two main drawbacks. The first is by virtue of its decision tree topology a Bayesian network does not allow for feedback paths, and the second is that obtaining the probabilistic information required for the network is not easy, especially for large networks.

Whilst Bayesian networks do not allow feedback paths between genes, a general graph does, and this manifests itself in the presence of cycles in the graph. Normally a cycle involves several nodes (genes) connected in such a way as to form a closed path in the graph, hence indicating a closed feedback regulatory system. Such cycles or repeated patterns in general are known as network motifs and they usually involve a few genes. A closed path implies directionality in that it has a start and an end and they coincide. It should be emphasised that a motif need not form a closed cycle; it is essentially a given pattern of interconnection of nodes that form a subgraph of the main graph. Motifs are sometimes referred to as modules since they can be viewed as performing separate functions in a modular fashion, as such they represent the next level of abstraction in studying gene regulation (figure 3-4).

A motif consists of a small number of genes with a certain interconnection pattern that results in specific dynamic behaviour of this small regulatory network. Two examples of possible interconnections of three genes are depicted in figure 3-5. Among all the possible motifs, only a subset occurs in real life gene networks as verified by experimental results. Alon (2007a) provides a comprehensive graphical illustration of all possible graphs for three and four genes, and gives in depth analysis

of some of the more common ones, in particular the three gene feed forward motifs, which can be either coherent or incoherent.

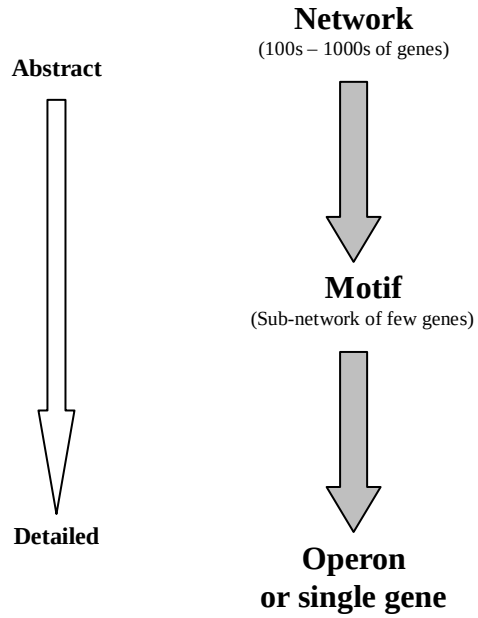
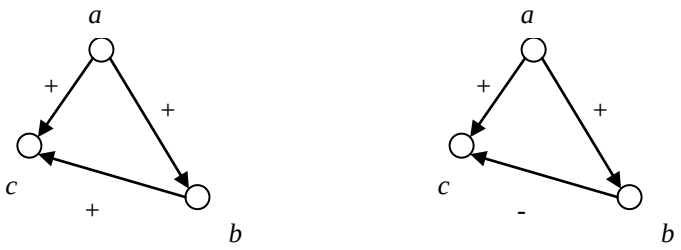


Figure 3-4: Levels of abstraction based on network size (hierarchy).

An example of each is depicted in figure 3-5, where the gene *a* affects both genes *b* and *c*; its effect on gene *c* is through two routes, a direct one, and an indirect one through gene *b*. The motif on the left is a coherent feedforward loop where the effect of gene *a* on gene *c* is the same through either the direct route *a*-*c* or the indirect route *a*-*b*-*c*, both activating *c*, hence coherent. The motif on the right is an incoherent feedforward loop where the route *a*-*c* activates *c* while the route *a*-*b*-*c* inhibits it. The mathematical formalism used for studying the dynamics of motifs is differential equations or Boolean networks as will be explained later.

Perhaps the most obvious, possibly controversial, assumption regarding motifs is their very existence. To be able to analyse motifs separately means that they are assumed not to interact with the rest of the network of which they are part, a notion that is questioned by some on the basis of potential cross talk between different motifs (Hartwell *et al.* 1999; Vilar 2006). The argument usually presented in support of the existence of motifs is drawn from metabolism where the metabolic network is

broken down into independent sub-networks each with a different function, such as glycolysis and the other different anabolic and catabolic pathways.



Coherent Feedforward Loop

Incoherent Feedforward Loop

Figure 3-5: Two possible motifs involving three genes *a, b* and *c*.

The + sign indicates activation of the gene at the head of the arrow by the one at its base, while the – sign indicates inhibition.

The lowest level of abstraction in the classification based on network size or equivalently on number of genes considered, is a single gene or operon. Recall from chapter two that an operon is a collection of functionally related genes that are co-expressed, and are found in bacteria. This level of abstraction is the one most used, since it is easier to obtain detailed experimental information and test hypotheses for a few genes than for a few hundreds. The related studies aim to understand the regulation of gene expression on a molecular level, i.e. how the different molecules involved affect expression of the gene investigated. Again the most common mathematical approach employed is differential equations for quantitative studies, and Boolean algebra for qualitative ones.

The above classification of the modelling approaches of the regulation of gene expression is based on network size, and it is clear that the lower the level of abstraction, the more detail about the individual genes is needed (figure 3-4). This is normally the case with other classifications as well, the more abstract we get the less detail we need and vice versa. In this vein, Ideker and Lauffenburger (2003) provide another classification that can be considered conceptual in the sense that it is based on the type of information utilised. Their classification is summarised in table 3-1 below. It should be noted that they are careful to point out that the demarcation lines between these levels are somewhat arbitrary in nature.

Table 3-1: Classification of gene regulatory models based on information utilised, according to Ideker and Lauffenburger (2003).

Level of abstraction	Information	Mathematical formalism
Highest (Abstracted)	Components & Connections	Statistical mining
		Bayesian networks
	Influences & Information	Bayesian networks
		Boolean models
	Mechanisms	Markov chains
Lowest (Specified)	Mechanisms including Structure	Differential equations

Schlitt and Brazma (2006, 2007) present yet another classification which is based on a mixture of functional and structural features of the regulatory system. Again they stress the two points mentioned above, namely the arbitrary nature of the division between the different levels of abstraction and that the amount of detail increases the less abstract the model becomes. They have suggested four levels of detail for modelling gene regulatory processes, summarised in table 3-2 below.

Table 3-2: Classification of gene regulatory models based on structural and functional information, according to Schlitt and Brazma (2007).

Amount of detail	Structure/Function	Purpose	Method/Mathematical formalism
Least	Parts list	Identify transcription factors and their targets	Bioinformatics and experimental
	Topology	Identify network topology or "wiring diagram"	Statistical tools
	Control logics	Identify regulatory effects (activation and inhibition)	Linear functions, Boolean functions Bayesian networks
Most	Dynamics	Describe and simulate dynamic response	Synchronous Boolean networks, Differential & difference equations

One can roughly see the correspondence between the concepts in the three classifications outlined above as summarised in figure 3-4 and tables 3-1 and 3-2. For example, the notions of Parts list and topology in table 3-2, components and connections in table 3-1 and network and motif in figure 3-4 are all related and convey structural information. Similarly, the concepts of control logics in table 3-2 and influences and information in table 3-1 relate to the signs and directions of

arrows in figure 3-5 (which is an elaboration of the middle tier of figure 3-4). Finally, dynamics in table 3-2 and mechanisms in table 3-1 relate to the mathematical formulations of regulation at the level of an operon or an individual gene as discussed earlier.

There are other classifications that are more or less of the same nature as the ones outlined above. Another classification worth mentioning however, because of its different perspective and its comprehensive nature is the mathematical classification presented by De Jong (2002). He surveys the different mathematical formalisms utilised in modelling the regulation of gene expression and what they can be used for. The classification criteria and the resulting classification are summarised in table 3-3 below. Three of those, namely static/dynamic, discrete/continuous and deterministic/stochastic have been discussed above. For the two additional classifications, qualitative/quantitative is self evident, while coarse/average/fine refers to the amount of detail that can be described by the corresponding formalism.

Table 3-3: Classification of gene regulatory models based on mathematical formalisms utilised and their properties, according to De Jong (2002).

Mathematical formalism	**Properties of mathematical formalism**				
	Static/ Dynamic	**Discrete/ Continuous**	**Deterministic/ Stochastic**	**Qualitative/ Quantitative**	**Coarse/ Average/ Fine**
Graphs	Static	N/A	Deterministic	Qualitative	Coarse
Bayesian Networks	Static	Both	Stochastic	Quantitative	Coarse
Boolean Networks	Dynamic	Discrete	Deterministic	Qualitative	Coarse
Generalised Logic Nets	Dynamic	Discrete	Deterministic	Qualitative	Average
Differential Equations (Linear or non-linear)	Dynamic	Continuous	Deterministic	Quantitative	Average/ Fine
Stochastic Master Equation	Dynamic	Discrete	Stochastic	Quantitative	Fine

In table 3-3, Generalised Logic Networks refer to networks where the variables can take more than two discrete values, hence they are a generalisation of Boolean

Networks where the variables take only two values. How this is done, and also the Stochastic Master Equation will be explained below.

Table 3-3 emphasises the point mentioned earlier about how the different features of a modelling method are independent of each other (e.g. e.g. dynamic discrete deterministic, etc). Perhaps the main drawback of this classification is that it does not say much about the situations in which these methods can be used. Granted, table 3-3 lists the features of each mathematical formalism, but unlike tables 3-1 and 3-2 above, it does not contain any information about the biological context. However, a modeller is normally expected to be familiar with the process he is planning to model and hence should be able to use this table as a guide for choosing the appropriate formalism.

We have presented several ways in which mathematical models of the regulation of gene expression can be classified. We now look at how the models themselves can be formulated. We will consider the two most commonly used formalisms, differential equations and Boolean functions. We will also consider them at the level of abstraction in which they are most widely applied, namely the modelling of a single gene or operon; thus corresponding to the bottom of figure 3-4 and of tables 3-1 and 3-2.

3.6 Modelling the regulation of gene expression using differential equations

Differential equations represent in general a function of the rate of change of a variable, in our context here with respect to time. Most of the models of gene expression at this level of abstraction entail modelling transcription regulation. As explained in chapter two, transcription of a gene takes place using the enzyme RNA polymerase, hence it can be viewed as an enzyme catalysed biochemical reaction in which the gene is the substrate and the mRNA molecule is the product. One way to study chemical reactions is using rate equations in which the rate of change of concentration of the chemical species concerned is studied in response to the

concentrations of other chemical species affecting the reaction. In a general form this is given by the differential equation

$$\frac{dx_i}{dt} = f(x_1,...,.x_j,...., x_n) \qquad 3\text{-}1$$

where the x_j's (with i and j = 1,2, ..., n) represent the concentrations of the different chemical species and the function f represents the dependence of x_i on all the x_j's, and is in general a non-linear function. In its simplest form, the reaction involves the conversion of one chemical species into another. Hence

$$\frac{dx}{dt} = f(x) \qquad 3\text{-}2$$

where x is the substrate, and naturally the rate at which it is consumed is the same as the rate the product is produced. For a simple enzyme catalysed reactions, the function f can be represented by the Michaelis-Menten kinetics, given by

$$\frac{dx}{dt} = \frac{V_{\max} x}{K_m + x} \qquad 3\text{-}3$$

where V_{max} is the maximum rate achievable, and it occurs when the enzyme is completely saturated with the substrate, i.e. x is very large. K_m is the substrate concentration at which the reaction rate is half the maximum. Both numbers can be verified by substituting in the equation above.

Michaels-Menten kinetics are derived based on the simplifying assumption that the enzyme essentially has one binding site and hence binds one molecule of the substrate. For an enzyme with more than one binding site, the binding of the substrate to one binding site may affect the affinity of the enzyme to the substrate (through the other binding sites). It can either increase the affinity to the substrate or decrease it, i.e. having either a positive or a negative effect respectively, known as a homotropic cooperativity. It may also increase the affinity to another molecule or decrease it, known as a heterotropic cooperativity.

For a protein with n binding sites, normally associated with n subunits of the protein, and full homotropic cooperativity, i.e. one substrate, the kinetics can be represented

by the Hill function. However, in its more general form, the Hill function can be formulated in a way that encompasses the more general case with different types of cooperativity.

$$\frac{dx}{dt} = \frac{V_{\max} x^n}{K_m^n + x^n} \qquad 3\text{-}4$$

where V_{max} and K_m have similar interpretations as above, and n is known as the Hill coefficient, which for the general case can take real values, i.e. not limited to integers. It is clear that when $n = 1$, the Hill kinetics reduces to the Michaels-Menten kinetics. The Hill function is depicted in figure 3-6 for different values of n.

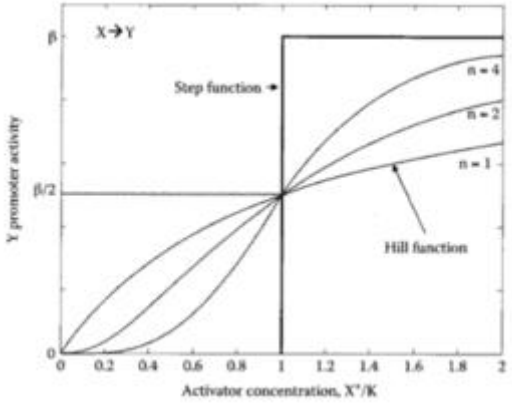


Figure 3-6: The Hill function for different values of *n*.

(Source: en.wikipedia.org, under the Creative Common Attribution Share License.)

When using the Hill function to model the regulation of the expression of a gene by an activator x, the left hand side represents the activity of the gene (the rate of its expression) in response to the change in concentration of the activator. When x is a repressor, the Hill function takes the form

$$\frac{dx}{dt} = 1 - \frac{V_{\max} x^n}{K_m^n + x^n} \qquad 3\text{-}5$$

which after normalisation and algebraic manipulation can be written as

$$\frac{dx}{dt} = \frac{V_{\max}}{1 + \left(\frac{x}{K_m}\right)^n} \qquad 3\text{-}6$$

In general the expression of a gene can be regulated by more than one molecule in particular transcription factors, in which case *f* becomes a function of more than one variable. Furthermore, the different variables relate to different genes, hence we end up with a network of genes being regulated by the same molecules, some of which are products of some of the genes involved. This situation can be represented by a set of coupled differential equations whereby for *n* variables we get

$$\frac{dx_1}{dt} = f_1(x_1, x_2, ..., x_n)$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, ..., x_n) \qquad 3\text{-}7$$

and

$$\frac{dx_n}{dt} = f_n(x_1, x_2, ..., x_n)$$

These equations can be written more concisely as a vector differential equation like below, where now $\boldsymbol{x}$ indicates a vector of variables and $\boldsymbol{f}$ a vector valued function of those variables and vector valued quantities represented in bold face

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}) \qquad 3\text{-}8$$

Which variables to include as factors affecting the expression of a gene will lead to different models for the same gene, and the same is true with the way those factors are assumed to exert their effects, as manifested in the function *f*. In effect this means that which molecules are included in the model and their assumed mechanisms of action will determine the model.

These crucial points can be demonstrated by the case of the *lac* operon explained in chapter two, which in spite of being well studied has a myriad of models describing its operation. A basic model would include glucose and lactose, however, the realisation that it is not those two molecules per se that effect the regulation, but others derived from them or using them, led to more complicated models. For example, allolactose - which is obtained by the cleavage of lactose - is the molecule that acts as an inducer of the *lac* operon rather than lactose itself. Similarly the operon

is activated by cAMP - which is produced in response to low glucose - rather than by glucose itself.

When considering the effect of lactose in the model, the first thing to do then is the choice of molecules, as the concentration of lactose outside and inside the cell will differ since inside the cell it will be broken down into allolactose. Furthermore, for this breakdown to take place we will not only need the enzyme β-galactosidase which cleaves lactose, but the enzyme galactoside permease as well, which allows lactose into the cell in the first place. This idea led to experiments with non-metabolisable inducers. For example Tian and Burrage (2005) take the different concentrations of the inducer inside and outside the cell into account and produce a model of six differential equations some of which are non-linear. A much simpler model of only four differential equations is produced by Vilar *et al.* (2003) again with emphasis on the permease. Santillan (2008) developed a comprehensive (and complicated) model that treats the operator region as three distinct locations (see chapter two) and considers how lactose interacts with them. He also considered the interaction of CAP with one of the operator sites, and included in the model the role of the sugars as energy sources rather than just inducers. This model has eighteen differential equations and twenty five parameters either estimated or obtained from the literature and was built on previous work by Ozbudak and others that involved extensive experimental investigations (Ozbudak *et al.* 2004; Santillan *et al.* 2007). Narang (2006) considers mixed substrate growth in a bioreactor, in particular that with glucose and lactose, and compares three models that take cell growth into account. A different approach is taken by Bintu *et al.* (2005) who formulate a model using statistical thermodynamics to find the probability of binding of RNA polymerase and the different transcription factors to the DNA molecule, and of the effector molecules to those transcription factors. Stochastic effects are also incorporated explicitly in the model by Stamatakis and Mantzaris (2009). The above is a very limited sample of models of the *lac* operon using differential equations, selected to illustrate the conceptual differences between them and is not meant to constitute an even limited survey of the topic.

Further complications of the modelling task arise in situations where molecules of a relevant chemical species are only present in a very low concentration. In such a case

the change by only one molecule out of a hand full can cause a large percentage change in concentration, hence the assumption of continuity on which differentiation is based would no longer be valid. Continuity is premised on the idea of an infinitesimal change which does not hold in such a case, and other methods are needed. One such method uses what is known as the Chemical Master Equation (CME), which like the statistical thermodynamics model mentioned above is based on the probabilities of two molecules coming in contact with each other and hence reacting. See Erdi and Toth (1989), and Wolkenhauer *et al.* (2004) for a discussion of these topics. Yet another complication comes from the fact that under normal conditions the cell is growing, hence effectively the volume in which the molecules are contained is increasing which means that the concentrations will be decreasing. Furthermore, the spatial distribution of the molecules in the cell is usually ignored, where the cell is treated as if it a well-mixed reactor, undermining the accuracy of the model further (Vilar *et al.* 2003).

What we want to demonstrate from this exposition is that a system as well studied and characterised as the *lac* operon, can still have a myriad of models that differ greatly in complexity. Even though all of these models are formulated at the same level of abstraction, that of the regulation of a single operon, they include different details. Such details are associated with the different molecules included in the model and the mechanism of their molecular interactions, and also the values of the parameters of the models. The fact that there are so many assumptions and uncertainties about the model and its parameters greatly undermines its value as a tool for quantitative analysis. Indeed, another investigation involving extensive experiments under different conditions and using two inducers namely, lactose and a non-metabolisable inducer, carried out by Setty *et al.* (2003) gave drastically different values for the two spanning orders of magnitude, but they did give the same trend. Hence the qualitative results gained from a model are arguably more reliable and consequently more useful than quantitative ones. This rationale is reflected in a quote by Fowkes and Mahoney (1994) where they state "*It is often the case that the qualitative insights gained from modelling are more important than any quantitative results obtained*."

In this section we have outlined how a quantitative modelling approach, namely differential equations can be applied, and demonstrated it conceptually using the *lac* operon. This discussion highlighted the problems with such methods which undermine their quantitative power and hence make the case for a simpler albeit only qualitative approach to modelling the regulation of gene expression. In the following section we will consider one of the most widely used qualitative methods, which is based on Boolean functions.

3.7 Modelling the regulation of gene expression using Boolean functions

The Boolean approach entails variables and functions that can take only two values, usually indicated by 1 and 0, hence termed binary. When applied to modelling the regulation of gene expression, these values can acquire corresponding biological interpretations. For example a gene can be either expressed or not expressed, i.e. ON or OFF, a regulatory protein (such as a transcription factor) either activated or deactivated, and an effector molecule (chemical species) either present or absent. This is an abstraction from the continuous case by which we assume that each of these biological entities can be in one of two states ignoring intermediate values, and hence can be represented by a binary variable. Note that the corresponding binary values do not in general refer to the actual physical values. For example when the variable representing the effector molecule is 1, this does not necessarily mean that its concentration is 1 (of whatever units used), it only means that the concentration is above the threshold needed to activate the protein. Similarly a value of 0 means that the concentration is below that which is necessary for activation.

Of course this is an approximation of the actual values of the variables, albeit a conceptually meaningful one. It can also be considered as a limiting case from a mathematical standpoint, where for example the Hill function in figure 3-6 approaches a Step function when the Hill coefficient *n* becomes very large. The Step function represents a binary variable in that it takes only two values, as evident from figure 3-6.

The Boolean approach can be used to model both static and dynamic relationships between the variables. By static we mean a fixed mapping between the inputs and the outputs, in the sense that the same inputs will always give the same outputs irrespective of the state the system is in. On the other hand in dynamic behaviour, the output will not only depend on the current input but also on the current state of the system. Those notions are somewhat different from the ones common in dynamical systems theory in which they would be referred to as autonomous or time-invariant, and non-autonomous or time varying respectively. The Boolean approach is extensively used in the design of digital electronic circuits, also known as logic circuits, and this is the point of view we will adopt here. For example the concepts raised above relating to static and dynamic behaviours have counterparts in digital circuits. Hence it will be helpful to give a brief overview of such circuits on an abstract level, i.e. not involving implementation details. This is standard logic design material that can be found in many textbooks on the topic, we have used Wakerly (2000) and Green (1986) which vary in treatment from the practical (Wakerly) to the more abstract (Green). We present the material here in an intuitive rather than a formal fashion and give examples that we believe make it more comprehensible to those without a background in electronic engineering.

3.7.1 Background on logic design

The terms Boolean, binary, logic and switching are often used interchangeably in electronic engineering, so are the terms circuit and network. The phrase logic circuit (or any of its variations from the terms above) normally refers to circuits whose elements can be in one of two states, either on or off, i.e. binary, such as switches. When expressed numerically as 1 and 0 respectively, such networks can be conveniently represented as Boolean functions and analysed using Boolean Algebra, whose operations are the well known logic AND, OR and NOT.

One way to classify logic circuits is into combinational and sequential circuits (figure 3-7). In a combinational logic circuit, the current output of the circuit is determined only by the combination of its current inputs. For a sequential logic circuit on the other hand, the current output depends in general on the current inputs and the current state of the circuit, which in turn is determined by the past sequence of inputs. Thus,

such circuits possess some form of memory functionality that retains previous states. They also employ some feedback mechanism to facilitate the utilisation of these states in determining the current output. Thus, in comparison with the discussion above, combinational circuits represent static behaviour while sequential ones represent dynamic behaviour.

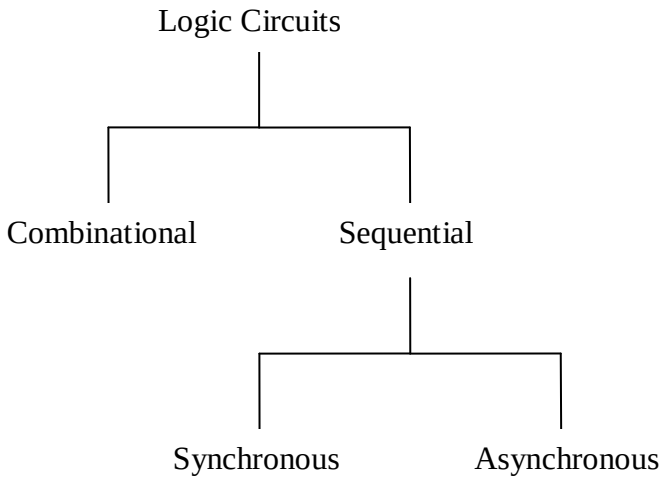


Figure 3-7: Classification of logic circuits.

We clarify the difference between the two types of circuits (or functions in general) by the example of a television remote control. When selecting a channel using the numerical keypad we go directly to the selected channel, for example pressing 5 on the keypad will take us to channel 5, and this will happen every time we press 5 irrespective of which channel we are currently watching. Hence the same input will always give the same output, thus representing a combinational logic function. Now consider pressing the up arrow on the remote control, the channel we go to will depend on which channel we are currently watching, which as mentioned above depends on how many times we have pressed the up and down arrows before. Hence which channel we will end-up at will depend not only on the current input (up arrow), but also on the current state (the channel we are currently watching), which in turn is determined by the past sequence of inputs, thus representing a sequential logic function. In short, the same input may give different outputs depending on the current state, and alternatively at a given state different inputs may give different outputs, which is the more common view. These concepts apply to the next state as well (not

just the current output) meaning that the next state of the circuit will be determined by the current input and the current state. Such type of sequential circuits is said to have the Markov property. However, in some cases we might need past states in addition to the current one to be able to determine the next state. A common example of such a sequential circuit is the traffic light controller. When the current state is yellow, information about the previous state - whether it was green or red - is needed in order to be able to decide on the next state.

It is clear that analysis and design of sequential circuits is more complicated than that of combinational ones. An additional layer of complexity is introduced when several such elements or circuits are interconnected, as the changes of their states may be synchronous (all occurring together) or asynchronous (figure 3-7). Further complexity is added when the occurrences of some conditions are random in nature; an example is when a car arrives at the sensor at an intersection to trigger the traffic light change. Such conditions need to be modelled probabilistically.

In the analysis and design of combinational circuits, the mapping of the inputs to the outputs is often represented by a truth table, which lists all the possible input combinations and the corresponding outputs. This is then expressed as a Boolean function which can be manipulated mathematically using the Boolean algebra rules to investigate the circuit behaviour. One of the common ways to express a Boolean function is the Disjunctive Normal Form (DNF), more commonly known as the canonical Sum of Products (SOP) to engineers. As the name implies, such an expression contains the sum of product terms, some of which may have negated variables. In Boolean algebra, a product term also known as a conjunction is an AND gate (or operator), a sum or disjunctive term is an OR gate and negation is a NOT gate. Hence any combinatorial circuit can be represented using those three types of gates (Birkhoff and Bartee 1970), the truth table representation of which is given in table 3-4 where the first two columns contain all the possible values of the variables x_2 and x_1, and the other columns contain the result of applying the operators to those variables.

Table 3-4: The logic operators (gates) AND, OR and NOT.

x_2	x_1	$x_2\ x_1$ (x_2AND x_1)	$x_2 + x_1$ (x_2 OR x_1)	$\bar{x}_2$ (NOT x_2)	$\bar{x}_1$ (NOT x_1)
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

Sequential circuits on the other hand have more ways of definition and representation than combinational ones. Common representations include a state transition table which relates the current output and next state to the current input and current state, and state transition diagram which is a compact pictorial representation of the state transition table. In addition to the logic gates just described, sequential circuits need components with memory functionality, those are known as flip-flops.

Given a combinational circuit represented by its truth table, (table 3-5), its canonical sum of product expression (or DNF) is given by the sum of all the possible product terms (also known as the min terms). Thus for a two variable function, the DNF is given by

$$f(x_2, x_1) = d_0\bar{x}_2\bar{x}_1 + d_1\bar{x}_2 x_1 + d_2 x_2\bar{x}_1 + d_3 x_2 x_1 \qquad 3\text{-}9$$

where the d_i's are binary constants taking the values 0 or 1 depending on the function specification, and the over-bar on the variables indicates logic negation (NOT). We will revisit this expression later in the following chapters.

Table 3-5: Truth table representation of a generic logic function.

min term number	x_2	x_1	$f(x_2, x_1)$
0	0	0	d_0
1	0	1	d_1
2	1	0	d_2
3	1	1	d_3

The reader might wonder why in the tables and equation we write x_2 on the left of x_1 rather than on the right. This is a matter of convention in which the juxtaposition of the variables is viewed as a multiple digit number, in our case this is x_2x_1, and the

variable with the smallest subscript (x_1 in this case) is placed in the least significant digit position followed by the next higher one to its left and so on.

Another matter of convention relates to the presentation of the values of the variables in the truth table, those are normally ordered in ascending order of binary count. This means that values of the variables when viewed as a multi-digit binary number, count in binary with the increasing row number. The equivalent count in decimal is given in the "min term number" column of table 3-5. We mention these two conventions here, namely order of the variables in the columns and order of their values in the rows, to avoid early on any distraction by the notation on the expense of the concepts.

3.7.2 Applying Boolean algebra to modelling the regulation of gene expression

Application of concepts from logic design to the modelling of the regulation of gene expression is best illustrated by an example. Consider the *lac* operon discussed in the previous chapter, in particular table 2-2 in chapter two describing its operation. It is repeated in table 3-6 below, with the terms High and Low referring to concentration substituted for by 1 and 0 respectively, similarly with the terms ON and OFF referring to the expression of the operon.

Table 3-6: Truth table representation of the *lac* operon.

Glucose x_2	Lactose x_1	Operon expression
0	0	0
0	1	1
1	0	0
1	1	0

It can be expressed in the disjunctive normal form by a straightforward substitution of the values of the d_i's in the equation to get

$$f(x_2, x_1) = \bar{x}_2 x_1 \qquad 3\text{-}10$$

This translates to the logic statement [(NOT glucose) AND lactose], which means that for the operon to be turned on (its genes expressed) we must have the condition

that there is no glucose and that there is lactose. This agrees with the explanation of the *lac* operon in chapter two. Hence representing gene regulatory functions using Boolean functions is a convenient and compact method that faithfully captures its behaviour (within the binary assumption of course).

This simple example illustrates a combinational gene regulatory function, sometimes referred to in the context of gene regulation as a *cis*-regulatory function (Yuh *et al.* 1998); more complicated functions are also investigated in the literature (Buchler *et al.* 2003). It should be noted that the term "combinatorial" is used in the gene regulation literature in place of the term "combinational" which is used in the logic design literature. Hence in the rest of this work we will adopt the former.

Sequential logic is used to model gene regulatory functions that involve interactions between the different genes forming regulatory networks as discussed above. There is a large body of research in this area including the early work of Kauffman and of Thomas among others, and is still an active area of research, (Glass and Kauffman 1973; Thomas 1973, 1991; Kauffman 1993; Faure *et al.* 2006).

The major advantage of Boolean models is that they are intuitive and are straightforward to formulate, especially for combinatorial functions. However, like any other modelling approach, they too suffer from some major shortcomings (Smolen *et al.* 2000). The most obvious of course is that they only allow for two states for the variables considered. Furthermore sequential logic models usually assume that all the states in the system will be updated synchronously, i.e. they will all change at the same time, which of course is not realistic. Also the qualitative behaviours described by the sequential logic models do not always correspond to the ones predicted by continuous models, in particular in the number of steady states (Smolen *et al.* 2000). Allowing more than two states and allowing for asynchronous state transitions should provide a better approximation to the continuous case.

Whilst allowing for only two states for a variable or a function might be a reasonable approximation in some situations, it is not so in others. One area of gene regulation where such an approximation is not accurate is in morphogenesis, which is the creation of shape and form in (usually) higher organisms. This is achieved by a

concentration gradient of chemicals generically known as morphogens where the highest concentration is at its point of generation and gradually decreases with distance from that point (Gilbert 2000). At different thresholds of this concentration, different genes are expressed. Hence it can be considered as a discrete variable but with more than two values, where the number of discrete values corresponds to the number of activation thresholds.

Thomas and co-workers addressed this problem by assuming a number of (dummy) binary variables that is the same as the number of thresholds, with each being 0 when the original variable is below a given threshold, and 1 when it is above it as depicted in figure 3-8 adapted from their paper (Snoussi and Thomas 1993). Hence for a variable with *m* thresholds, we get *m* binary variables dividing the range of that variable into *m* + 1 regions.

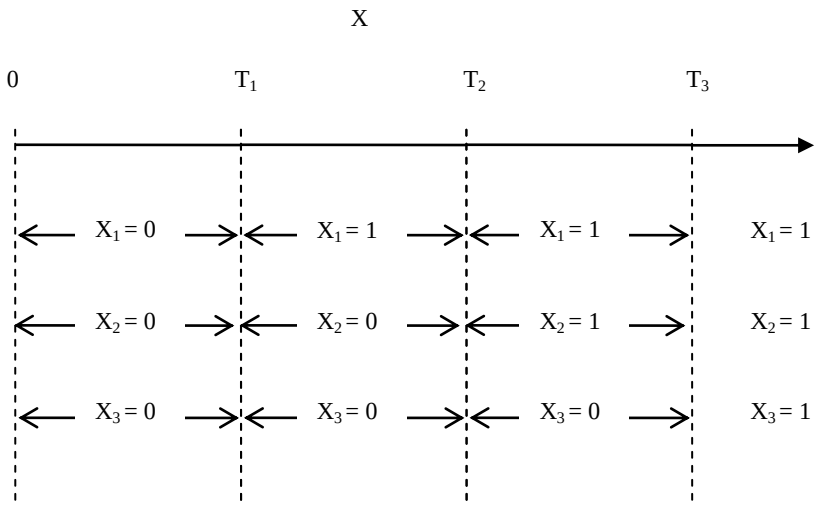


Figure 3-8: A multiple-valued discrete variable represented as a number of binary variables.
Adapted from Snoussi and Thomas (1993)

In figure 3-8, the continuous variable X has three thresholds denoted by T_1, T_2 and T_3, which result in three binary variables denoted X_1, X_2 and X_3 dividing the range of the original variable X into four activation regions. When using such an approach in modelling combinatorial logic functions, we get an unnecessarily large number of awkward variables that do not have intuitive meaning and that make both the formulation and analysis more tedious. There are other methods for modelling discrete multiple-valued functions that are normally used in the modelling of discrete

event systems, and include Petri nets and finite automata among others which are mainly applied in computer science rather than engineering (Cassandras 1993). Petri nets in particular are gaining wider applicability in modelling gene regulation (Comet *et al.* 2005; Chaouiya *et al.* 2008), however they get exceedingly complicated with scale (Murata 1989).

We will thus develop a multiple-valued state modelling approach that naturally extends the use of Boolean models beyond two states, albeit with some minor restriction on the number of states. The method is based on the algebraic structures known as finite fields, also known as Galois fields, where the binary representation is a special case of the multiple-valued one. We will also give it other mathematical and biological interpretations. Hence we first need to introduce some concepts from abstract algebra, which is the topic of the next chapter.

Before moving to the next chapter, we need to highlight some caveats common to all gene regulatory models whether qualitative or quantitative. Firstly for multi-cellular organism (Eukaryotes), most of the models in the past were based on measurements *in-vitro*, i.e. when the cell is not within its organism, yet they are claimed to represent the situation *in-vivo*, i.e. when the cell is within the organism. Recent advances in analytical techniques are starting to tackle this anomaly. Secondly for single cell organisms (Prokaryotes and Eukaryotes), measurements are made on communities of cells, yet they are assumed to represent the situation for each individual cell, an assumption that is questionable on the basis that the measurements actually represent an ensemble (spatial) average (Vilar *et al.* 2003; Wolkenhauer *et al.* 2004). Furthermore, when modelling regulation at some level, say transcription, the effect of molecules at other regulatory levels e.g. translation is ignored (see figures 2-6 and 2-7 in chapter two). Whilst sometimes justified by the difference in the speed of response of the different molecular processes, known as the quasi-steady state assumption, it does nonetheless undermine the accuracy of the models. Indeed there is a growing trend towards multi-scale modelling where the effects of the processes at more than one scale are integrated in one model (Arnold *et al.* 2005).

3.8 Summary and Conclusion

This chapter covered two main topics, the first relates to modelling in general and the second to the modelling of the regulation of gene expression in particular.

In the first part we introduced some theoretical concepts underlining modelling and some practical considerations to take into account when building models. From this part we concluded that the accuracy of a model is constrained by both the simplifications of the system behaviour in order to build the model and the mathematical formalism used to build it. Both of those issues are determined by the decisions a modeller makes when building a model, such as deciding on the level of abstraction of the treatment and the amount of detail to include in the model in addition of course to the mathematical formalism to use. In this part we also briefly discussed different classifications of models. We also gave a brief synopsis of the evolving role of mathematical modelling in biology. The main points from the first part of this chapter are summarised at the end of sections 3.2 and 3.4.

The second part of this chapter was concerned with applying the concepts introduced in the first part, to the modelling of the regulation of gene expression. In particular we classified models based on different criteria such as the scale of the genetic network being modelled, the functions and information involved and other hybrid criteria, and demonstrated a loose equivalence between the different classifications. Up to this part all the treatment was conceptual in that it did not contain equations, the mathematical nuts and bolts. The main ideas are summarised in figure 3-4 and tables 3-1, 3-2 and 3-3. In the remainder of the second part of this chapter we presented two types of models quantitative and qualitative. For each we discussed the most common method, namely differential equations and Boolean functions respectively and discussed their application to the *lac* operon, the well studied bacterial regulatory system. This demonstrated both the advantages and drawbacks of each method. Differential equation models have the advantage of providing quantitative information, however, this is undermined by the many uncertainties involved in building the model. These include the choice of molecules to include in the model and their mechanisms of interactions, and the unavailability of some parameters which thus have to be either estimated from the data or assumed. In addition, they often ignore spatial distribution

of the molecules and delays in responses. These shortcomings call for the use of qualitative models among which Boolean functions are the most popular. However, in spite of their simplicity and intuitive appeal Boolean functions also suffer from shortcomings, most notably that they are limited to only two values for the variables. This has prompted us to develop a method (to be presented in chapter five) for which the binary case is a special case of the multiple-valued one.

Thus the outcome to take forward from this chapter is that there is a need for another method for modelling gene regulatory functions that can represent them as discrete multiple-valued functions. This method is based on concepts from abstract algebra and functional analysis. Hence we introduce those in the next chapter.

Chapter 4: Algebraic Structures

4.1 Introduction

The purpose of this chapter is to introduce two mathematical concepts that will be utilised in later development of a discrete modelling formalism for the regulation of gene expression, namely finite fields and vector spaces. We have indicated in the previous chapter that the concentration of effector molecules that activate proteins and the activation states of those proteins, in addition to the expression levels of genes can all be modelled qualitatively by discrete states. The simplest case is two states leading to the Boolean formalism, but they can also have more than two states requiring a different mathematical approach. In this chapter we introduce the mathematical background needed for developing such an approach.

The presentation philosophy here mimics that in chapter two. In chapter two when we discussed the regulation of gene expression, we started by introducing some fundamental concepts in cell biology and then built the other concepts of the chapter on them. Here too, we will start by reviewing some fundamental concepts in algebra such as a set and a binary operation with its different properties, which we will then use to introduce some of the common algebraic structures. An algebraic structure is essentially a set with one or more binary operations defined on it the properties of which determine the resulting algebraic structure. We will briefly examine algebraic structures with one binary operation and discuss a representative example namely groups. We will also consider algebraic structures with two binary operations mainly fields, with finite fields as a representative. We will then use fields to introduce vector spaces which are abstractions of the familiar Euclidian space; we will abstract further by considering function spaces. We are mainly interested in discrete structures here, but most of the treatment applies to the continuous case as well.

Many of the concepts discussed in this chapter are already familiar to engineers, albeit in a less abstract form. Hence, just as chapter two was an introduction to gene expression for the non-biologist, this chapter is an introduction to abstract algebra for

the non-mathematician. Since this work is presented to an engineering school, engineers are ultimately the intended readership. Consequently our approach in presenting the material here will be an intuitive one that demonstrates the concepts by concrete examples rather than by theoretical proofs and prolonged derivations common in mathematical treatment. Thus most of the sources used for this chapter are engineering ones and hence related to applications rather than theory. Among those is the book by Davio *et al.* (1978) which covers most of the material in this chapter, but is unfortunately not very readable and does indeed have such a reputation in the literature. Another book that has been consulted frequently when writing this chapter is that by Gallian (1994). Other books that have also been useful are those by Birkhoff and Bartee (1970), Rosenbrock (1970), Naylor and Sell (1982), and Strang (1988). It is important to remember that the purpose here is not to present the mathematical results for their own sake, but in order to utilise them in later development.

4.2 Some fundamental concepts in algebra

Modern algebra, also known as abstract algebra because of its abstract approach, deals with sets of objects and binary operations on those sets, unlike classical algebra which is concerned with numbers and formulas and the arithmetic operations on them (Birkhoff and Bartee 1970). Modern algebra is also concerned with abstract algebraic structures and their properties, both of which are in essence abstractions of the more common algebraic notions related to numbers. Hence the concepts presented here are abstractions of those that most engineers are already familiar with.

One of the most fundamental concepts in abstract algebra is that of a set, which is merely a collection of entities. The members of the set - known as its elements – do not necessarily have to represent numbers, as they can indicate any abstract elements that usually share some common property. For example the set of genes or proteins or regulatory functions and so forth. The elements of a set can be discrete, termed countable, such as the set of integers, or may form a continuum and hence uncountable, such as the set of real numbers. The number of elements in a set, known as its cardinality, can be finite or infinite. Clearly only countable sets can have a finite number of elements.

A binary operation can be defined on a set, whereby binary means that it acts on two elements of the set at the same time. It is essentially a rule that involves the two elements to give another element. Examples of binary operations on integers and real numbers include addition and multiplication. If the outcome of the binary operation is also an element of the set, the set is said to be closed under this operation; such property is termed the closure property. As an example, the set of integers is closed under multiplication, but not under division because the quotient of the division of two integers is not always an integer, and hence will not belong to the original set. It is worth mentioning that the notion of closure is often embedded in the definition of a binary operation. In such a case a binary operation is defined as a function that assigns to each pair of elements of the set an element of the same set. According to this definition, addition is considered a binary operation on the integers while division is not. In our treatment here, however, we will consider closure as a property rather than as part of the definition, as we believe this will make it easier to explain some of the algebraic concepts involved.

Depending on the properties that the set has under the given binary operation, one gets different algebraic structures (Gallian 1994). Such properties are well known and include the associative property, the existence of an identity, the existence of inverses, and the commutative property. As a review of those properties, consider the elements a, b and c to be any arbitrary elements belonging to the set of real numbers, and consider the binary operation to be normal addition. Then those properties can be described as follows

1. Closure property

   $a + b$ is an element of the set

2. Associative property

   $a + (b + c) = (a + b) + c$

3. Existence of an identity (denoted by 0)

   $a + 0 = 0 + a = a$

4. Existence of an inverse for an element a of the set

   There exists an element, often denoted "$-a$" such that

   $a + (-a) = (-a) + a = 0$, the identity of addition

5. Commutative property

   $a + b = b + a$

We have used addition only as an instance of a binary operation and the set of real numbers as an example of a set. The properties have the same definitions for other binary operations and other sets, although possibly with different notations.

The reader might be tempted to ask "So what? Don't those properties always hold?" It is here where we need to resort to abstraction to appreciate that this is not necessarily the case. For example, it was pointed out above that the set of integers is not closed under the binary operation of division. Similarly, the associative property does not hold for division for the set of real numbers as demonstrated by the quotient 10/(5/2) = 10/(2.5) = 4, which is different from the quotient of (10/5)/2 = 2/2 = 1. For the identity property, consider the set of even numbers under normal multiplications, it does not have a multiplicative identity because there is no number in the set that satisfies property 3 above (for multiplication). For the existence of an inverse, consider the set of say 2 × 2 matrices under matrix multiplication, not all such matrices have an inverse as singular matrices will not. Similarly matrix multiplication is not commutative. The key point in these counter examples is to consider the idea of a set and a binary operation in a wider sense than that of the usual set of real numbers under the normal arithmetic operations.

The more properties a set has under the operation considered the progressively richer the algebraic structure gets. It is possible to define two binary operations on the same set in which case even richer structures emerge. We will consider in the next sections structures with one and with two binary operations,

4.3 Groups

In this section we consider algebraic structures with a single binary operation defined on them, in particular structures known as groups, but we will have a brief look at simpler structures first.

Among the five properties of a binary operation outlined above, when only the first two are met, the algebraic structure is known as a Semigroup (Rosenbrock 1970). When the third property is met as well, we get a Semigroup with identity, more commonly known as a Monoid (Birkhoff and Bartee 1970), (table 4-1). If the first

four properties are all met, the algebraic structure is known as a Group, in such a case the identity is unique and furthermore, each element of the Group will have a unique inverse. Groups constitute one of the most important algebraic structures and we will discuss them further here. It should be noted that when the binary operation is also commutative, i.e. property 5 is met, we get commutative versions of each of the structures above, namely a commutative semigroup, a commutative monoid and a commutative group. The term Abelian is often used interchangeably with the term commutative, in which case we say for example an Abelian Group.

Table 4-1: Algebraic structures with one binary operation.
An x indicates that the property applies.

Algebraic structure	Property			
	Closure	Associative	Identity	Inverse
Semigroup	x	x	-	-
Monoid	x	x	x	-
Group	x	x	x	x

There are myriad examples of groups such as the set of real, rational or complex numbers under addition, all of which represent Abelian groups. Those sets are continuous; however discrete sets also form groups such as the set of integers under addition. Similarly the set of real, rational and complex numbers but without zero represent Abelian groups under multiplication whereby the identity is 1, and where zero is excluded because it does not have a multiplicative inverse. However, the set of integers (excluding zero) is not a group under multiplication because the multiplicative inverse of an integer is not an integer (except for 1). If we abstract from these common examples, we find that the set of all square matrices of dimension two for example form an Abelian group under addition. They do not form a group under multiplication however, because singular matrices will not have an inverse (they behave like zero does for real numbers). If the set is restricted to non-singular matrices then we get a group, but it is not Abelian because matrix multiplication is not commutative as has been indicated in the previous section.

The above examples involved number systems or matrices, however, as mentioned earlier abstract algebra is not limited to numbers but addresses different types of objects. To demonstrate this, we now give an example adapted from Gallian (1994) to illustrate the above concepts for a more abstract case. Figure 4-1 depicts a square

followed by its clockwise rotation by the angles 0^o (i.e. no effect), 90^o, 180^o and 270^o from its original position, as indicated by the positions of the four letters A, B, C and D near its corners. Each rotation is denoted by the letter R and the angle of rotation as a subscript. Now form the set S = $\{R_0^o, R_{90}^o, R_{180}^o, R_{270}^o\}$ and define the binary operation as the composition of two elements of the set, i.e. the outcome of the binary operation of two consecutive rotations is the resultant rotation.

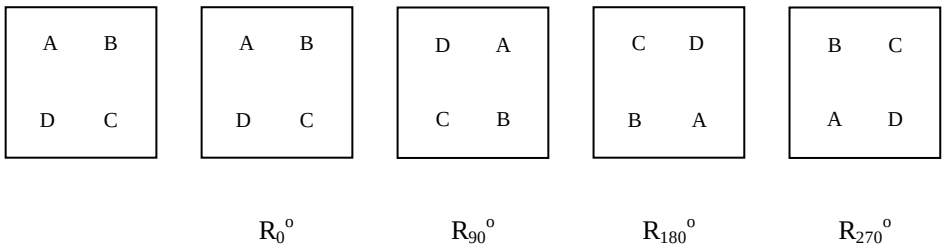


Figure 4-1: A square and the effect of its rotations by the angles 0^o, 90^o, 180^o and 270^o from its original orientation.

It is clear from figure 4-1 that the identity of this binary operation is the rotation R_0^o as it does not affect the outcome of any other rotation, e.g. $R_0^o R_{90}^o = R_{90}^o$. It can also be verified from the figure that R_{90}^o and R_{270}^o are the inverses of each other because they cancel out the effect of each other, i.e. the inverse undo what the original operation does. So a rotation by 90^o followed by a rotation by 270^o results in a rotation by 360^o which is equivalent to the original figure without rotation, i.e. R_0^o, the identity of the binary operation. Also each of R_0^o and R_{180}^o is the inverse of itself, hence every element of the set has an inverse. The associative property can also be verified by inspection. Thus the set S with the binary operation as defined does indeed form a group. In fact it is an Abelian group since the resultant of any two rotations does not depend on the order they are carried out. This group is known as the Cyclic Rotation group of the square in the plane.

The binary operation on the elements of the set is commonly represented in a tabular form known as the operation table or Cayley table, where the operation is indicated in the top left hand corner of the table and the top row and left column contain the elements of the set. The table shows the outcome of the binary operation on the elements of row *i* and column *j* in the table. The operation table for the example above is given by table 4-2, where for example R_{180}^o (third row of the rotations)

followed by $R_{90}{}^{o}$ (second column of the rotations) yields $R_{270}{}^{o}$ (the cell at the intersection of the relevant row and column). Note that the order is important since not all groups are Abelian. For the particular group in the example above, examination of table 4-2 demonstrates all the properties of an Abelian group, namely properties 1 to 5 in the previous section.

Table 4-2: Operation table for the cyclic rotation group of the square.

Composition of Rotations	$\mathbf{R_0{}^o}$	$\mathbf{R_{90}{}^o}$	$\mathbf{R_{180}{}^o}$	$\mathbf{R_{270}{}^o}$
$\mathbf{R_0{}^o}$	$R_0{}^o$	$R_{90}{}^o$	$R_{180}{}^o$	$R_{270}{}^o$
$\mathbf{R_{90}{}^o}$	$R_{90}{}^o$	$R_{180}{}^o$	$R_{270}{}^o$	$R_0{}^o$
$\mathbf{R_{180}{}^o}$	$R_{180}{}^o$	$R_{270}{}^o$	$R_0{}^o$	$R_{90}{}^o$
$\mathbf{R_{270}{}^o}$	$R_{270}{}^o$	$R_0{}^o$	$R_{90}{}^o$	$R_{180}{}^o$

We can extend the set S to include reflection H around a horizontal axis passing through the midpoint of two sides, a similar reflection V around a vertical axis and a reflection around each of the diagonals denoted D_1 and D_2. We again define the binary operation as the composition of two operations, any two of the different reflections and rotations. We still get a group, known as the Dihedral group of the square (also known as its group of symmetry), although now this is not Abelian because not all the elements commute. For example a reflection followed by a rotation is in general not the same as a rotation followed by a reflection; the reader can verify this by attempting such compositions on figure 4-1. The two groups described here are instances of a more general class of groups known as the groups of symmetry of lattices which are commonly used in crystallography and stereochemistry.

This example was particularly chosen to demonstrate several concepts. It is meant as an exercise in abstraction, to free the mind from restricting algebraic structures to numbers and mathematical operations. As a consequence, it showed the elements of the set as rotations, but they may as well be anything else, for example electronic components, genes or proteins, with the binary operation defined according to the context. Furthermore, it demonstrates that the set underlying a group can have a finite number of elements and not necessarily infinite like the set of integers. This last point serves as a motivation for the idea of modular arithmetic discussed next.

4.3.1 Modular arithmetic

Following our presentation philosophy, we will start with a concrete example and use it to extract the general ideas. Consider a finite set of integers say the set of four integers {0, 1, 2, 3}, if we use ordinary integer addition as the binary operation we will get numbers that do not belong to the set, i.e. the set will not be closed under normal addition. For example whilst 2 and 3 are elements of the set, their sum 5 is not. To overcome this anomalous situation we re-define addition using what is known as modular arithmetic. In modular arithmetic an integer is represented by the remainder of its division by some number (also an integer), and the normal arithmetic operations are performed with respect to that number. In the case of our example here this is performed with respect to the number of elements in the set, i.e. four, thus any number greater than 3 will be represented by the remainder of its division by 4. For example 4 will be represented by 0 because there is no remainder for the division, 5 will be represented by the remainder 1 since 5/4 = 1 + 1/4, and so on.

Formally an integer *a* divided by another integer *m* gives a quotient *q* and a remainder *r* (we restrict our discussion to positive integers), that is

$$\frac{a}{m} = q + \frac{r}{m} \qquad 4\text{-}1$$

where the remainder *r* will always be less than *m*. The above equation can be rewritten as

$$a = qm + r \qquad 4\text{-}2$$

This form is used to define the modular representation of integers, which means that an integer *a* modulo another integer *m*, is represented by an integer *r* which is the remainder (or residue) of the division of *a* by *m* and denoted by

$$a(\mathrm{mod}\, m) = qm + r = r \qquad 4\text{-}3$$

a is said to be congruent to *r* modulo *m*. It follows that all integers with the same remainder are considered equivalent from the point of view of modular representation.

This representation can be used to perform arithmetic operations on finite sets of integers, where addition and multiplication are now defined using the modular representation of the elements of the set. In such a modular arithmetic we have

$$a(\text{mod } m) + b(\text{mod } m) = (a + b)(\text{mod } m)$$
$$a(\text{mod } m) \times b(\text{mod } m) = (a \times b)(\text{mod } m) \qquad \text{4-4}$$

When we consider the example of the set {0, 1, 2, 3} above, we find that now with modular representation 2 + 3 gives 1 (since 2 + 3 = 1 mod 4) which is an element of the set. This is indeed the case for all the elements of the set as indicated in table 4-3, hence the set is now closed under modular addition.

Note that a similar notion of modularity is used in clock representation whereby the hours are calculated from the minutes modulo 60. A demonstration of that is in bus schedules where bus arrival times are indicated by a certain minute after the hour, irrespective of what the hour is, hence considered equivalent. Note also that the idea of re-defining arithmetic operations to suit a different context is not totally alien to engineers since it is used in defining the multiplication of complex variables for example, where the phase in addition to the magnitude is involved in the multiplication.

Table 4-3: Operation table for addition modulo 4 on the set {0, 1, 2, 3}.

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

A close inspection of table 4-3 reveals that the set in question, under modular addition actually forms an Abelian group. The five necessary conditions are met, closure has just been demonstrated, associativity and commutativity are evident by inspection of table 4-3, and the identity is 0, only the inverses might not be obvious. To clarify that, when we look at every row (or column) of the table we find that the identity 0 appears once, the two elements at whose intersection this occurs are inverses of each other. From table 4-3 we can see that 1 and 3 are inverses because 1 + 3 = 3 + 1 = 0 mod 4, while 2 is its own inverse. Further inspection of table 4-3 indicates that it is identical

in structure to table 4-2, the operation table of the cyclic rotation group, except for a change of notation, i.e. the elements of the set and the binary operation have different names, but same properties. This is a further demonstration of the power of abstraction in that it reveals the commonality among markedly different applications.

As a matter of notation, Z_m refers to the set of the first m integers $\{0, 1, \ldots, m\text{-}1\}$, with addition and multiplication defined modulo m. Whilst this set is always a group under modular addition, this is not always the case under modular multiplication. For example table 4-4 shows the multiplication table for the set {0, 1, 2, 3} discussed above. When considering a multiplicative group we always exclude 0 because it has no multiplicative inverse. When we consider the rest of the elements of the set we find that the multiplicative identity 1 does not appear in the row or column of 2 indicating that 2 has no multiplicative inverse. Thus the set fails one of the conditions for a group, and hence is not a group. The exception for this case is when the number m is a prime number (a number divisible only by 1 and itself), in which case the set Z_m (excluding zero) forms a group under modular multiplication, since in such a case every element will have an inverse. We will not go into the details of this here.

Table 4-4: Operation table for multiplication modulo 4 on the set {0, 1, 2, 3}.

×	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

It is clear from this discussion that the properties relate to the binary operation and are not inherent to the set, since the same set may form a group under one operation and not under another.

4.4 Finite Fields

When two binary operations are defined on a set, the above properties will need to apply to each of them separately, moreover the relation between the two operations introduces an additional level of richness to the algebraic structure. Depending on

which properties are met by the binary operations and how they relate to each other we get different algebraic structure, just as with the case of a single binary operation. Consider a set with two binary operations defined, that we will generically call addition and multiplication, bearing in mind that this is only a name and does not necessarily mean that they correspond to the usual addition and multiplication. We can have several cases regarding the properties of the two operations, however we will only consider the two most important ones: a Ring and a Field.

For a set to constitute a ring under the two binary operations addition and multiplication - irrespective of how they are defined - the following conditions must be fulfilled (recalling that zero is always excluded when considering multiplication)

- The set forms an Abelian group under addition
- The set forms a semigroup under multiplication
- Multiplication is distributive over addition, i.e. for *a*, *b* and *c* elements of the set, we have
  $a.(b + c) = a.b + a.c$

When the semigroup is commutative we get a commutative ring. Examples of rings include the set of real, rational and complex numbers under normal addition and multiplication. Since a ring does not require a multiplicative inverse for elements of the set, it follows that the set of integers under the same operations is a ring, also the set of square matrices of a given dimension under matrix addition and multiplication is a ring, albeit a non-commutative one. Another very important type of rings is the polynomial ring, i.e. the set whose elements are polynomials and where the binary operations are polynomial addition and multiplication. The ring of integers and the ring of polynomials have corresponding properties and roles especially in building finite fields. The examples just mentioned have infinite number of elements, however there are rings with a finite number of elements as well, such as the set Z_m with modular addition and multiplication discussed above. The set Z_4 whose operation tables are presented in table 4-3 and 4-4 above is an example of the finite case. The multiplication table of the set, table 4-4, reveals one of the problems with rings namely that they can have a zero divisor. This means that there can be two non-zero elements in the set whose product is zero, this is the case with the number 2 as is clear from table 4-4. This anomalous situation restricts the usefulness of rings, in particular

in solving algebraic equations. Hence there is a need for a more powerful algebraic structure that alleviates this problem, and this is the role of fields.

For a set to constitute a field under addition and multiplication (however we define them), the following conditions must be fulfilled:

- The set forms an Abelian group under addition
- The set forms an Abelian group under multiplication
- Multiplication is distributive over addition as explained above

With their additional properties fields enable solving algebraic equations, and of particular importance among those are polynomials. The simplest polynomial is that of degree one, i.e. a linear equation given by

$$ax + b = c \tag{4-5}$$

where *a*, *b*, *c* and the possible values of *x* all belong to some field

To be able to solve this equation, we need first to have an additive inverse for *b*, which we will indicate here by (-*b*), to get

$$ax + b + (-b) = c + (-b) \tag{4-6}$$

$$ax = c + (-b) \tag{4-7}$$

We then need a multiplicative inverse for *a*, that we will indicate by a^{-1}, to get

$$a^{-1}ax = a^{-1}[c + (-b)] \tag{4-8}$$

$$x = a^{-1}[c + (-b)] \tag{4-9}$$

and hence obtain the value of *x*.

This simple example demonstrates why fields with their property of the existence of both additive and multiplicative inverses for their elements, are needed for the solution of linear equations. We have also used the associative property with both addition and multiplication.

Note that in the above analysis we did not place any conditions on the field in question, hence it applies to any field. Whilst the sets of real, rational and complex

numbers under normal addition and multiplication all form fields, the set of integers does not as it does not form a group under multiplication as discussed earlier. But how about a set with a finite number of integers, for example the set Z_m discussed above, does it form a field under modulo m addition and multiplication? The answer depends on the value of the integer m. We have demonstrated above that Z_m is always an Abelian group under modular addition so this part of the definition of a field is fulfilled. As for multiplication we have indicated that Z_m is a group only when m is a prime. It can also be proved that modular multiplication is distributive over modular addition. Hence we can conclude that a set with a finite number of elements is a field under modular addition and multiplication when this number is a prime. Such a field is known as a Galois field and denoted by GF(p) where p is a prime number which is the number of elements of the field (also known as the order of the field), and where the operations on the field are defined modulo p. The definition of a finite field also applies when the order of the field is a power of a prime, in which case the field is denoted by GF(p^n) where n is a positive integer. GF(p^n) is known as the extension field of the base field GF(p), but now addition and multiplication for the extension field are defined differently from the modular definition of the base field.

In general a finite field is denoted by GF(q) where q can be either a prime or a positive integer power of a prime. The elements of the field and the operations on them are defined using polynomials on the field, but we will not elaborate on that here as the treatment becomes too abstract and bears no direct impact on the remainder of this work. Finite fields and polynomials on them are used extensively in coding theory, part of communication engineering, we refer the interested reader to some of the classic texts in that area which we have used, for example (Berlekamp 1968; Lin and Costello 1983; McEliece 1987).

We have demonstrated the theoretical importance of fields, in particular their fundamental role in algebra and that is in solving equations, but why are they important to us here, in the context of this work? There are two reasons for that, both relate to our application of modelling the regulation of gene expression. Of special interest to us in this context are finite fields and in particular functions defined on them. Before we discuss those however, we need to briefly review another fundamental concept in mathematics, that of a function.

A function in its basic form is a mapping from one set called the domain of the function to another set called its codomain, and is denoted by f: A → B, where A is the domain and B the codomain. The function assigns to each member of the domain a single element in the codomain. The domain and codomain can be the same set, in which case we get f: A → A. Engineers are familiar with the concept of functions where the domain and codomain are usually the set of real numbers, so we get functions such as $f(x) = x^3$, or $f(x) = \sin x$. However, in line with our abstract approach in this chapter, the sets need not represent number systems and the function does not have to be a mathematical formula. For example consider the set of students in a class as the domain of the function; define a function that assigns each student to a gender, so the codomain will be the set of two elements {Male, Female}, another function on the same set of students can assign to each student a nationality category say {UK, EU, International}.

The domain of a function can comprise more than one set, for example we can then define a function that determines the funding options for a student based on their gender and nationality category. This last function would have those two sets as its domain and the set of funding options as its codomain, indicating that the function can map the product of several sets, known as the Cartisian product of those sets, to a single set. When we map the Cartisian product of two sets X and Y to a third set Z, we indicate this by f: X×Y → Z, where members of the domain are represented by (x, y) known as an ordered pair. The reader may have deduced where we are heading with this line of thought, and that is a binary operation as defined above is essentially a mapping from the Cartisian product of a set by itself, to itself (when the operation is closed), denoted by f: X^2 → X. The adjective "ordered" in the phrase "ordered pair" is important because not all operations are commutative, hence the function may give a different value for (x, y) than for (y, x). Note that the notion of a binary operation can be generalised to an n-ary operation which is an operation involving n members of a set (or in general members from n sets).

Functions can be defined on any algebraic structure including the ones outlined above. Furthermore, we can have a set whose elements are functions and whose binary operations are defined accordingly. For example we can define the binary operation as function composition as we have effectively done with the rotation group of the

square above. This is common in mathematics and loosely means that the codomain of one function is used as the domain of the other function. It is interesting to note that when those functions map a set to itself, then the set of such functions under the binary operation of function composition forms a monoid, an algebraic structure discussed above. The identity of the monoid in this case is the identity function which maps each element of the domain to itself.

Now we return to our motivation for studying fields, and the first reason for discussing them here, is restricted to finite fields and relates to functions defined on them. One of the powerful properties of a finite field is that any function defined on it can be represented by a polynomial on the field of degree less than the order of the field.

A polynomial $f(x)$ of degree n over a field F is given by

$$f(x) = \sum_{k=0}^{n} a_k x^k \qquad 4\text{-}10$$

$$= a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \qquad 4\text{-}11$$

where the coefficients a_k and the values that x can take belong to F, hence the values of $f(x)$ will also be in F. This is a powerful property that we will use in our development of a method for modelling the regulation of gene expression in the next chapter.

It is worth mentioning at this stage that Boolean algebra which was discussed in the previous chapter in the context of qualitative modelling of the regulation of gene expression, is also an abstract algebraic structure consisting of a set with two binary operations. Those are the usual AND and OR operators sometimes referred to as Boolean product and Boolean sum respectively. One of the properties of these operations that may appear unfamiliar to some engineers is that they are both distributive over each other thus giving (for a, b and c Boolean values)

$$a.(b + c) = a.b + a.c \qquad 4\text{-}12$$

$$a + (b.c) = (a + b).(a + c) \qquad 4\text{-}13$$

Another counterintuitive property is that known as idempotency, given by

$$a + a = a \qquad 4\text{-}14$$

In addition Boolean algebra has a unary operation, i.e. an operation acting on only one element, which is the operation of complementation denoted by NOT. Such a structure is known as a distributive complimented lattice. We will not go through further details here, but the interested reader can follow the theoretical aspects of Boolean algebra in Birkhoff and Bartee (1970) and the applied aspects, especially to logic design in Wakerly (2000).

We have mentioned earlier that there are two reasons why we are interested in fields in this work; the first has to do with functions on finite fields. The second reason is their role in defining vector spaces, which we explore further in the next section.

4.5 Vector Spaces

A vector space is a generalisation of the concept of the three dimensional Euclidean space, albeit not limited to three dimensions. As is well known to all engineering students, a vector has a direction and a length. The direction of the vector is determined by its co-ordinates; an n-dimensional vector $\boldsymbol{v}$ will have the co-ordinates $(a_1, a_2, ..., a_n)$, where the a_k's belong to some field. Note that, as mentioned earlier, the order of the a_k's is important, for example the three dimensional vector (a, b, c) will in general have a different direction from (a, c, b) even though they will both have the same length. The length of a vector can be scaled by a factor, appropriately known as a scalar and belongs to the same field as the coordinates.

The key point to note from this description of a vector, which is not normally stressed in engineering courses, is that the numbers representing the coordinates and the scalar belong to a field in the general sense explained in the previous section. Indeed, this field is not limited to the field of real numbers as is commonly practised in engineering courses, but can be any field as discussed above including a finite field, with the mathematical operations being those of the field considered. This leads us to the issue of what sorts of operations can we perform on a vector, and consequently what sort of algebraic structure emerges?

In line with the abstract approach employed in this chapter, we introduce an algebraic structure known as a vector space. As with any other structure it consists of a set with binary operations defined on it. In this case we have a set V of vectors, where each vector $\boldsymbol{v}$ consists of an n-tuple, i.e. an ordered sequence of n numbers $(a_1, a_2, ..., a_n)$ from some field F as just mentioned. The elements of F are known as scalars and the binary operations of F apply to them. We also define two binary operations on the set V

- Vector addition defined as the component-wise addition of two vectors. This binary operation forms an Abelian group on the set V.
- Scalar multiplication defined as multiplying a scalar by every component of the vector, and satisfying the following properties were a and b are scalars and $\boldsymbol{u}$ and $\boldsymbol{v}$ are vectors (as a matter of convention, letters at the beginning of the alphabet indicate scalars and those towards the end indicate vectors and presented in bold face)
  1. $a\,(b\,\boldsymbol{v}) = (a.b)\,\boldsymbol{v}$
  2. $a\,(\boldsymbol{u} + \boldsymbol{v}) = a\,\boldsymbol{u} + a\,\boldsymbol{v}$
  3. $(a + b)\,\boldsymbol{v} = a\,\boldsymbol{v} + b\,\boldsymbol{v}$
  4. $1\,\boldsymbol{v} = \boldsymbol{v}$

  It should be stressed that this operation involves a scalar and a vector, unlike the well known operation of the dot product of two vectors also commonly referred to as scalar multiplication.

In an abstract sense, and using the notation introduced earlier, vector addition is a mapping f: $V \times V \rightarrow V$, and scalar multiplication is a mapping g: $F \times V \rightarrow V$

Note that in stating the above properties we have ignored some of the subtleties involved, where we used the same addition symbol for two different operations, namely vector addition of property 2 and addition in the field F of property 3. For multiplication we used the dot for the multiplication of two scalars as on the right hand side of property 1, and juxtaposition for the multiplication of a scalar and a vector as on its left hand side.

Since the vector space is closed under vector addition and scalar multiplication, it follows that the sum of any number of (scaled) vectors, known as their linear combination, is also a vector in the space. Conversely it can be proved that there is a

set of vectors $\{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_n}\}$ in the vector space, whereby any vector $\boldsymbol{v}$ in the space can be represented as a linear combination of these vectors, i.e. it can be expressed as

$$\boldsymbol{v} = c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2} + ... + c_n\boldsymbol{v_n} \qquad 4\text{-}15$$

where the c_k's are scalars belonging to the field F.

Such a set is said to span the vector space, and when it is the smallest set (i.e. the one with the least number of vectors) with this property, it is known as the basis of the vector space. An important condition on this set is that its vectors are linearly independent, meaning that they can never be linearly combined to give zero. In other words, we can not have

$$c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2} + ... + c_n\boldsymbol{v_n} = \boldsymbol{0} \qquad 4\text{-}16$$

unless all the c_k's are zero

A basis set is the maximal linearly independent set in the space, meaning that any set with more vectors will not be linearly independent. It is also the minimal spanning set, meaning that any set with fewer vectors will not span the entire vector space. The size of the basis set, i.e. its number of elements is the dimension of the vector space.

As an aside, to avoid confusion we highlight a matter of notation in the use of brackets. The parentheses or round brackets (,) are used to enclose the components or coordinates of a vector, while the braces or curly brackets {, } are used to enclose elements of a set, whatever that set is. So for example, the n-tuple $(a_1, a_2, ..., a_n)$ represent the components of an n-dimensional vector, while $\{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_m}\}$ represents a set of m vectors, each consisting of an n-tuple as above.

Recall that any vector in the Euclidean space can be represented as a linear combination of its coordinates indicating that they are linearly independent. But they also have an additional property and that is they are perpendicular, more formally known as orthogonal. It follows that any set of orthogonal vectors is linearly independent but not the vice versa. The condition for orthogonality is well known to

engineers, basically that any two vectors are orthogonal if their dot product (or more generally the inner product) is zero.

The inner product is a binary operation on the set of vectors that assigns a scalar value from the underlying field F to the product of two vectors, i.e. it is a function given by h: Vx$V \rightarrow F$. We will not go through the formal properties of the inner product. One example of an inner product is the dot product, and for two real valued vectors $\boldsymbol{a} = (a_1, a_2, ..., a_n)$ and $\boldsymbol{b} = (b_1, b_2, ...,b_n)$ is given by

$$\boldsymbol{a}.\boldsymbol{b} = \sum_{k=1}^{n} a_k b_k \qquad 4\text{-}17$$

As with the other algebraic structures discussed above, a vector space is an abstract structure as well, i.e. it does not have to correspond to a geometrical space, but can represent any other space. For example, consider organic molecules consisting of the three elements Carbon, Hydrogen and Oxygen, if we imagine they form a basis set of a vector space, then they will span this space as depicted in figure 4-2 (Palsson 2006). This means that any compound consisting of those three elements, such as a carbohydrate, will fall in this space and will be represented by a linear combination of them.

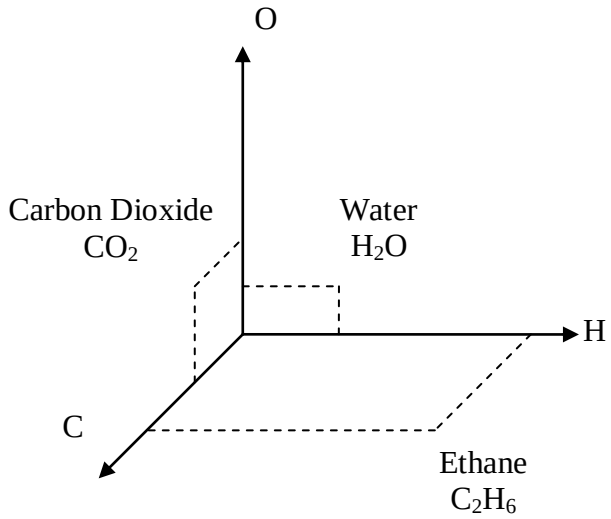


Figure 4-2: A vector space of the three elements Carbon, Hydrogen and Oxygen.

Note that this space represents the elemental composition only, hence it is known as the elemental space (Palsson 2006). It does not provide any information on the stereochemical structure of the organic molecules in the space, thus all molecules of the same composition will be represented by the same point in this space. We have included three elements so that we can visualise it, however, in principle we can include more elements in which case we can represent more compounds in such a vector space. As an extreme case, the set of all hundred or so elements in nature will span the space of all substances in the world! Again this demonstrates the power of abstraction in enabling using the same tools for approaching markedly different problems.

An excellent source for engineers on vector spaces and linear algebra in general is the book by Strang (1988). It should be noted that an inner product is not required for the definition of a vector space as outlined above. A vector space is an algebraic concept, while the concept of orthogonality which is based on an inner product is a geometric one (Naylor and Sell 1982).

We can define functions from one vector space to another as we can do with any other algebraic structure, of particular interest is a class of functions known as linear transformations. Consider two vector spaces X and Y over some field F, and define a function L: $X \rightarrow Y$ such that for any vectors $\boldsymbol{x_1}$, $\boldsymbol{x_2}$ and $\boldsymbol{x}$ in X, and scalar a in F we have

$$L(\boldsymbol{x_1} + \boldsymbol{x_2}) = L(\boldsymbol{x_1}) + L(\boldsymbol{x_2}) \qquad 4\text{-}18$$

$$aL(\boldsymbol{x}) = L(a\,\boldsymbol{x}) \qquad 4\text{-}19$$

this function L is a linear transformation from X to Y. Any linear transformation on a vector space can be represented by a matrix. Linear transformations are very important in algebra and have many engineering applications especially in functional analysis, which we introduce very briefly next.

4.5.1 Functional Analysis

We have discussed above vector spaces with dimension n wherein a vector can be represented by an n-tuple of points from some field F. The question now is what

happens if those points become infinite? In this case the vector space is considered infinite dimensional and every vector will have an infinite number of points. When those points form a continuum, they can be considered as functions and we get a function space. These functions can be defined on any set *S*, the domain, but their values will be from the field *F* of the vector space. So a vector in this space will be defined by a function f_i: $S \rightarrow F$, where *S* is the domain and *F* the co-domain of each function.

The concepts of linear independence and span and hence the concept of a basis will carry over to the infinite dimensional case. We mention two examples of a linear combination of vectors on a function space, namely the Taylor series and the Fourier series.

A continuous and infinitely differentiable function $f(x)$ can be represented by its Taylor series expansion which all engineers are familiar with and is given by

$$f(x) = f(0) + f'(0)x + f''(0)\frac{x^2}{2!} + \ldots \qquad 4\text{-}20$$

Or in a concise form

$$f(x) = \sum_{k=0}^{\infty} c_k x^k \qquad 4\text{-}21$$

where the c_k's are calculated from the derivatives of $f(x)$ and divided by the factorials as in the equation above. This is a linear combination of single term polynomials x^k (sometimes referred to as monomials). Those polynomials are linearly independent because their linear combination cannot be identically zero (i.e. zero for all values of x) unless all the coefficients c_k's are zero. Hence the set of polynomials form a basis for functions fulfilling the conditions mentioned. The expansion of course is valid only in the region of the definition of the function where the series converges.

The next example is the Fourier series expansion which many engineers are familiar with. For a periodic function $f(t)$ with period ω_0, the Fourier series is given by

$$f(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos k\omega_0 t + b_k \sin k\omega_0 t) \qquad 4\text{-}22$$

This can also be expressed in exponential form as

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t} \qquad 4\text{-}23$$

where the coefficients a_k and b_k or alternatively c_k are calculated from the well known Fourier integrals.

This is a linear combination of the trigonometric functions, or alternatively the exponentials. Again those functions are linearly independent, furthermore, they are orthogonal because the inner product of any two sines (or cosines) with different frequencies is zero. In fact, with the appropriate scaling the functions can be made orthonormal, recall that an orthogonal set of vectors where each vector has unit length, known as a unit vector, is called an orthonormal set. It should be noted however, that now the inner product has to be defined according to the vector space at hand, so in this case it is defined using integration.

Many other function spaces can be defined with different domain set and co-domain field for the functions involved. In particular, the set on which the functions are defined can be a finite set and the field on which it takes its values can be a finite field, which is the case we are interested in for the remainder of this report. Of course the corresponding basis, inner product and transformations will be specified according to the functions involved. The study of such spaces is known as functional analysis, more commonly known to engineers are operator theory. An excellent source for engineers on this topic is the book by Naylor and Sell (1982).

4.6 Summary and Conclusion

The treatment in this chapter can be thought of as consisting of two braided strands constituting knowledge and skill. The first is the mathematical knowledge presented while the skill that we hope we have managed to develop an appreciation for and familiarity with is that of abstraction. Admittedly this chapter might be somewhat difficult to read. This is probably not as much due to the difficulty of the mathematical subject matter, as it is due to the abstract view of the concepts involved.

In presenting the mathematical subject matter we started with the very basic concepts and moved to more advanced ones. Mathematical concepts are often thought of as being built on each other in a tower of Babel fashion, therefore we followed a strategy by which each concept was built on the previous ones. So we used the concept of a single binary operation on a set to introduce groups, which we then used together with two binary operations on a set to introduce fields. Fields were then used to introduce vector spaces which were abstracted further to present function spaces.

Some topics might appear not to have direct impact on our work such as groups and other concepts. However, groups for example are there for two reasons relating to the two strands of our presentation. Firstly they provided an introduction to the other algebraic structures that followed them, and secondly as an exercise in abstraction that helps to build the intuition into an application.
It should be noted that the treatment here was very simplified and we consciously avoided many of the theoretical details as our purpose in this chapter was the results and not how they were arrived at. Consequently we deliberately avoided proofs and derivations.

There are two main mathematical concepts that we want to take forward from this chapter, namely fields and vector spaces. Among the fields, finite fields are of particular interest to us within the context of this work for two reasons. Firstly, because of their powerful property by which any function on a finite field can be represented by a polynomial on the field. Secondly because they can be used to model finite sets and hence are suitable candidates for modelling multiple-valued biological variables such as concentrations of molecules, activation states of proteins or expression levels of genes.

The second concept is that of a vector space which we have indicated is based on a field. Again of particular interest are vector spaces based on finite fields because they will be used in our modelling of the regulation of gene expression. Another notion related to vector spaces that was introduced here and that will be used in future development is the idea of a function space, which is essentially a space whose vectors are functions. We will be interested in discrete functions in particular, hence weaving the two main topics of this chapter a function on a finite field and a space of

such functions into the mathematical fabric of a vector space of functions on a finite field.

We will use all these concepts in the next chapter to model the regulation of gene expression.

Chapter 5: Algebraic Modelling of Combinatorial Gene Regulatory Functions

5.1 Introduction

In this chapter we present another modelling approach for combinatorial gene regulatory functions that results in an equation known as the Reed-Muller expansion of the function. We have indicated in the introduction to this work in chapter one that application of a new method to an existing problem should provide an advantage on current methods for investigating the problem. The method mentioned here allows the problem to be seen from a different perspective, and allows investigating other related problems. Using the concepts introduced in the previous chapter we will give the Reed-Muller expansion three different algebraic interpretations, each of which will give biological insight and useful tools that enable investigating different problems related to gene regulation.

This is the main contribution of this work and it migrates concepts across disciplines in such a way that allows posing the problem of one discipline in the form of another problem in a different discipline. One of the particularly interesting applications we present below is posing the problem of detecting mutations in the genome of an organism as the problem of detecting a fault in an electronic circuit. Perhaps the intellectual contribution here is in the ability to detect and formulate the commonality between the two seemingly different problems. This entails abstracting and detaching the domain specific details of a problem from its domain independent core, allowing one to see the commonalities between problems across domains.

We have indicated above that we will investigate three algebraic interpretations of the Reed-Muller expansion. The first interpretation is to view the Reed-Muller expansion as a function on a Boolean algebra, but in a way different from that mentioned in chapter three. The second is to view it as a polynomial on a finite field, and the third as a transformation on a function space, both of those were introduced in the previous

chapter. All three will have different biological interpretations and use. To make the material manageable we will cover the first two in this chapter and the third will be treated separately in the next chapter.

5.2 The Reed-Muller Expansion

It has been pointed out in chapter three that the levels of gene expression can be abstracted to two states corresponding to maximum and minimum expression or on and off; similarly with the different variables involved in gene regulation such as the activation states of a protein and the concentrations of effector molecules. Such binary (i.e. two valued) variables are normally represented by the values 1 and 0 respectively, i.e. they belong to the set {0, 1}. Consequently the regulatory functions defined on these variables can be modelled using the rules of Boolean algebra.

Here we will take a different approach to modelling these functions, based on the fact that the set {0, 1} with the appropriate definition of addition and multiplication constitute the finite field GF(2). Given that 2 is a prime number, then addition and multiplication can be defined modulo 2, with their operation tables shown in table 5-1. The number 2 is actually the smallest prime number (other than unity), consequently GF(2) is the smallest finite field. In fact it is the simplest since it consists of only two elements, the minimum required of a field, namely the additive identity 0 and the multiplicative identity 1. It follows then from the closure requirement of a field that the only non-zero element in the field which is 1, is its own additive and multiplicative inverse. This makes arithmetic operations on GF(2) particularly simple as is evident from table 5-1.

Table 5-1: Addition and multiplication modulo 2.

+	**0**	**1**
0	0	1
1	1	0

×	**0**	**1**
0	0	0
1	0	1

Again as mentioned in chapter three when a binary function f of n binary variables denoted by f: $\{0, 1\}^n \rightarrow \{0, 1\}$ is considered as a function on a Boolean algebra (whose operations are the logical AND, OR and NOT), then it can be represented by

the disjunctive normal form (DNF). Recall that the terms binary, Boolean and logic are often used interchangeably.

On the other hand when the function f is considered as a function on the finite field GF(2) whose operations are addition and multiplication modulo 2, then it can be represented by another canonical form known as the Reed-Muller (RM) expansion. This representation has its origins in the early work of Reed and of Muller separately, on error correcting codes and on Boolean functions (Reed 1954; Muller 1954), although there are claims that it had been developed earlier in the former Soviet Union and Japan separately (Falkowski 1999; Stankovic and Sasao 2001). As is common in our approach throughout this work, we will explain the RM expansion using a concrete example then extend it to the general case subsequently.

Consider a two variable function specified by the truth table in table 5-2, repeated below from chapter three where the numbering convention and the definition of min term were explained. The d_i's in the table are binary constants taking the values 0 or 1 according to the value of the function at the corresponding values of the variables x_1 and x_2.

Table 5-2: Truth table representation of a generic logic function.

min term number (m)	x_2	x_1	$f(x_2, x_1)$
0	0	0	d_0
1	0	1	d_1
2	1	0	d_2
3	1	1	d_3

The RM expansion of this function is given by (Green 1986; Almaini 1994)

$$f(x_2, x_1) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_2 x_1 \qquad 5\text{-}1$$

where the encircled sum symbol (also known as the ring sum) denotes modulo 2 addition and juxtaposition of variables denotes modulo 2 multiplication, as it does in other multiplication.

There are several algorithms in the literature for computing the coefficients a_i's and are primarily concerned with computational efficiency (Habib 1993; McKenzie *et al.* 1993; Falkowski and Rahardja 1997). Here however, we are more interested in the analytical rather than the numerical side of the problem as it gives interesting insight into its biological interpretation. We will thus use a method presented by Green (1986) and also by Almaini (1994).

To find the coefficients a_i's, we substitute the different values of the variables x_1 and x_2 and the corresponding values of the function from table 5-2 into equation 5-1 to get

$$\begin{aligned} d_0 &= f(0,0) = a_0 \\ d_1 &= f(0,1) = a_0 \oplus a_1 \\ d_2 &= f(1,0) = a_0 \oplus a_2 \\ d_3 &= f(1,1) = a_0 \oplus a_1 \oplus a_2 \oplus a_3 \end{aligned} \qquad 5\text{-}2$$

This can be put in matrix form to give

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \qquad 5\text{-}3$$

These equations compute the function values d_i's from the coefficients a_i's. Normally however, the d_i's are given as they define the function, and we want to find the RM expansion. Thus rearranging equations 5-2 by merely adding the first equation to the second and to the third, and the first three to the fourth, then solving for the a_i's bearing in mind that we are using modulo 2 arithmetic, we get the RM coefficients in terms of the truth values of the function (Green 1986; Almaini 1994)

$$\begin{aligned} a_0 &= d_0 = f(0,0) \\ a_1 &= d_0 \oplus d_1 = f(0,0) \oplus f(0,1) \\ a_2 &= d_0 \oplus d_2 = f(0,0) \oplus f(1,0) \\ a_3 &= d_0 \oplus d_1 \oplus d_2 \oplus d_3 = f(0,0) \oplus f(0,1) \oplus f(1,0) \oplus f(1,1) \end{aligned} \qquad 5\text{-}4$$

Or in matrix form as

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \qquad 5\text{-}5$$

Note that equations 5-4 and 5-5 have the same form as equations 5-2 and 5-3 respectively.

We have mentioned that the RM expansion, like the DNF is a canonical representation of a function. In this context, a canonical representation is one that can uniquely express every possible function of the variables. For *n* binary variables there are 2^n combinations of values; when those are used as inputs to a binary function, we get 2^{2^n} different possible output functions. For $n = 2$, there are 4 ($=2^2$) different combinations of the inputs and 16 ($=2^4$) possible functions with truth values shown in table 5-3.

Table 5-3: All possible binary functions of two binary variables.

m	Inputs		Outputs															
	x_2	x_1	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

To clarify the numbering notation in table 5-3, we recall from chapter three that the order of the inputs x_2x_1 is chosen such that the variable with the lowest index corresponds to the least significant digit of a binary number. We also use the same convention when numbering the functions where we arrange them such that the index of a function represents the decimal equivalent of a four digit binary number whose most significant digit is f (1,1) and least significant digit is f (0,0). Thus the index represents the binary number given by f (1,1) f (1,0) f (0,1) f (0,0) which ranges from 0000 to 1111 with equivalent decimal value ranging from 0 to 15 indicating the 16 different functions in table 5-3. As an example to clarify this numbering notation, consider the function f_7, which has the values f (1,1) = 0, f (1,0) = 1, f (0,1) = 1, f (0,0) = 1. When written as a single binary number this becomes 0111 which has the decimal value 7, the index of the function.

We know from equation 5-1 that the RM expansion has four different terms each with a coefficient a_i that can be either 1 or 0, hence we can have 2^4 (=16) different equations each with a unique combination of coefficients and hence of terms, corresponding to one of the sixteen functions in table 5-3. This intuitive argument demonstrates that the RM expansion uniquely represents every function on the field, and hence is canonical.

5.3 Combinatorial gene regulation as a function on a Boolean algebra

The Reed-Muller expansion of a binary function is expressed in terms of the operations of addition and multiplication modulo 2, given in table 5-1 above. A closer look at the table reveals that addition modulo 2 is the same as the exclusive OR (or XOR) logic operator in table 5-4. XOR gives a value of 1 when only one of its two inputs is 1 but not both, unlike the logic OR operator which gives 1 when either or both of its inputs are 1 (table 5-4). Hence the relationship between the two operators OR and XOR is given by

$$x_1 + x_2 = x_1 \oplus x_2 \oplus x_2 x_1 \qquad 5\text{-}6$$

where the plus sign on the left hand side indicates OR. The last term (x_2x_1) serves to eliminate the case when both variables are 1 simultaneously. Table 5-1 also shows that multiplication modulo 2 is equivalent to the logic AND operator. The two logic operators AND and XOR are presented in table 5-4 below.

Table 5-4: The logic operators AND, XOR, OR and NOT.

x_2	x_1	$x_2\, x_1$ (x_2AND x_1)	$x_2 \oplus x_1$ (x_2XOR x_1)	$x_2 + x_1$ (x_2OR x_1)	$1 \oplus x_1$ (1 XOR x_1)	$\bar{x}_1$ (NOT x_1)
0	0	0	0	0	1	1
0	1	0	1	1	0	0
1	0	0	1	1	1	1
1	1	1	0	1	0	0

Comparison of the two tables (5-1 and 5-4) reveals the equivalence between the GF(2) operators of addition and multiplication on the one hand, and the Boolean operators of XOR and AND on the other. Furthermore, it is well known, and also obvious from

table 5-4 that the complement of a logic value can be computed using the XOR operator by

$$\bar{x}_1 = 1 \oplus x_1 \qquad 5\text{-}7$$

As has been mentioned earlier, a binary function can be represented on a Boolean algebra by its disjunctive normal form (DNF) given by (see chapter three)

$$f(x_2, x_1) = d_0 \bar{x}_2 \bar{x}_1 + d_1 \bar{x}_2 x_1 + d_2 x_2 \bar{x}_1 + d_3 x_2 x_1 \qquad 5\text{-}8$$

where the d_i's are the values of the function as specified by its truth table, table 5-2 above. Now using equations 5-6 and 5-7 in equation 5-8, and using the distributive property of AND over XOR and the Boolean algebra rule stating that the logic product (i.e. AND) of a variable by its complement is 0, we get

$$f(x_2, x_1) = d_0(1 \oplus x_2)(1 \oplus x_1) \oplus d_1(1 \oplus x_2)x_1 \oplus d_2 x_2(1 \oplus x_1) \oplus d_3 x_2 x_1 \qquad 5\text{-}9$$

After simplification this becomes

$$f(x_2, x_1) = d_0 \oplus (d_0 \oplus d_1)x_1 \oplus (d_0 \oplus d_2)x_2 \oplus (d_0 \oplus d_1 \oplus d_2 \oplus d_3)x_2 x_1 \qquad 5\text{-}10$$

which is the RM expansion of equation 5-1 with the coefficients given by equation 5-4. This provides a derivation of the RM expansion without resorting to the finite field properties, but only using Boolean algebra; it is actually a canonical representation since it is uniquely derived from a canonical representation (the DNF). It also demonstrates that the RM expansion can be viewed as a Boolean function that uses XOR and AND rather than OR, AND and NOT. As an aside and on a more technical note, we do not even need the full properties of a Boolean algebra, a Boolean ring is sufficient, hence the term ring sum mentioned above. A discussion of rings is given in chapter four, however we will not pursue this line of thought here as it is too technical for our purpose, and will not provide any additional insight into the problem.

5.4 Biological interpretation of the Reed-Muller expansion

Representation of binary gene regulatory functions in the literature is mostly based on Boolean algebra in particular the disjunctive normal form (DNF) and to a lesser extent the related conjunctive normal form (CNF). The biological interpretation of the DNF in particular has been studied extensively in the literature. Among the good treatments of the topic are the works of Hwa related to generic regulatory functions (Buchler *et al.* 2003), Alon for *E.coli*, (Setty *et al.* 2003; Mayo *et al.* 2006; Kaplan *et al.* 2008) and Davidson for higher organisms (Istrail and Davidson 2005; Oliveri *et al.* 2008).

Boolean algebra is based on the logical operators AND, OR and NOT which are linguistic based connectors that combine logic statements, hence the terms disjunctive and conjunctive. They have their origins in formal logic in particular the so called prepositional logic (Rautenberg 2006). A DNF statement essentially lists the set of conditions whether positive (asserted) or negative (negated) that has to exist simultaneously for an outcome to occur. In chapter three we gave a Boolean expression for the lactose operon in the form of the logic statement "Operon expression = [(NOT glucose) AND lactose]", which can have the logic values - also known as truth values - of "True" or "False". Its logical interpretation is that when both conditions, the negative one and the positive one are True, then the outcome will also be True. In biological terms this means that when (NOT glucose) is True AND lactose is True, then the Operon expression is true. In other words when there is no glucose and at the same time there is lactose, then the operon genes are expressed. This representation and the associated interpretation is intuitive and can be manageable for a small number of variables, however, it becomes unwieldy, awkward and difficult to interpret when the number of variables is large. Furthermore, it lacks the analytical and computational power of the familiar mathematical manipulations on fields, even when the truth values of True and False are expressed numerically as 1 and 0, partly because of some of the unusual mathematical properties of Boolean algebra such as idempotency, see chapter four.

We have demonstrated that the RM expansion provides an equivalent mathematical representation to the Boolean expressions, in the sense that it can express all the

functions that can be represented by the Boolean expressions. We now consider the interpretation of the RM expansion in the context of gene expression regulation.

Because the RM expansion (see equation 5-1 above) is formulated in terms of the true variables, i.e. does not contain negated variables, the gene expression level can be calculated right away by substituting the values of the different variables. Consequently, when all the variables are zero, the RM expansion gives the basal expression level of the gene. Since a_0 is the only coefficient in the expansion that is not multiplied by any of the variables (i.e. the x_i's), we can tell right away by inspection the normal unregulated (basal) level of expression of the gene, without need for substituting any values for the variables. So what sort of biological interpretation of the RM expansion do we get when we substitute values for the different variables representing the regulatory factors affecting the expression of a gene? As usual we will consider concrete cases from which to generalise.

In their preliminary study of the transcription regulatory network of *E. coli*, Thieffry *et al.* (1998) studied 500 regulated genes from which they found that more than 300 were regulated by a single transcription factor, about 150 by two factors and the rest by three or four factors, with only one regulated by six factors. A more comprehensive recent study reflected the same pattern (Martinez-Antonio *et al.* 2008) known as a power law relationship whereby the number of genes regulated is inversely proportional to the number of regulating transcription factors (usually raised to some power greater than one) (Christensen *et al.* 2007). This means that the number of genes (or operons) regulated by two transcription factors is proportional to 1/2 while those regulated by three is proportional to 1/3, which is a lower number (of course multiplied by some factor), and so on. The situation is even more pronounced when those numbers are raised to some power greater than one. The importance of regulation by two factors is clear in the study by Kaplan *et al.* (2008) of nineteen sugar metabolism operons in *E .coli*. In light of this discussion we first consider the biological interpretation of the single variable RM expansion which is the most common case, and then consider the two variable case which is the second most common. It should be noted that the situation in yeast is more complicated and can have up to a dozen or more factors regulating a gene (Lee *et al.* 2002), and even

further complicated in higher organisms, in particular in the regulation of development (Davidson *et al.* 2002; Oliveri *et al.* 2008).

5.4.1 One variable regulatory function

For a one variable regulatory function the RM expansion is given by

$$y = a_0 \oplus a_1 x \qquad 5\text{-}11$$

There are four ($= 2^{2^1}$) possible binary functions for one variable obtained by the different combinations of the binary values of the coefficients a_0 and a_1, (table 5-5). As explained above, a_0 represents the basal transcription level of the gene which is then modulated by the regulatory factor x. If a_0 is 0 then the gene is normally off (not expressed) and x is an activator that when it becomes high (e.g. high concentration) turns the gene on. On the other hand if a_0 is 1 then the gene is normally on in which case x is a repressor that turns the gene off when it (the repressor) becomes high. Note that we are mainly interested in non-degenerate functions, i.e. those that depend on all the variables, in this case only one variable. The degenerate case occurs when a_1 is 0 leading to two trivial (unregulated) cases. The first case is when a_0 is 1 corresponding to a constitutive gene which is always expressed, such as housekeeping genes (see chapter three). The second case is when a_0 is 0 which is meaningless.

Table 5-5: Different biological explanations for the one variable Reed-Muller expansion. Note that a_0 and a_1 are the coefficients in the expansion while x is the input.

a_0	a_1	Equation	Explanation
0	1	$y = x$	x is an activator
1	1	$y = 1 \oplus x$	x is a repressor
1	0	$y = 1$	Constitutive gene (trivial)
0	0	$y = 0$	Meaningless case (trivial)

This simple example illustrates the benefit of the RM expansion as a modelling approach; we were able to infer the behaviour of the regulatory function solely by looking at the coefficients of the equation. For the single variable case we do not even need to substitute any values for the variable x. The different cases are summarised in table 5-5.

5.4.2 Two variables regulatory function

For a two variable regulatory function the RM expansion is given by

$$y = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_2 x_1 \qquad 5\text{-}12$$

As explained earlier, there are sixteen different binary functions for the two binary variables (table 5-3). Two of these functions are trivial, namely f_0 and f_{15} in table 5-3, and four functions are degenerate i.e. depend on only one of the two variables; those are f_3, f_5, f_{10} and f_{12} in table 5-3. The degenerate cases reduce to the single variable case discussed above. The remaining ten functions depend on both variables, and those are the ones we are interested in in this section. Again a_0 represents the basal expression level of the gene (or operon), so we will examine the other three coefficients. To avoid degeneracy both variables must appear in the equation, which leads to five cases for each of the two values of a_0 as analysed below.

Assume that a_0 is zero, i.e. the gene is normally unexpressed, then with regard to equation 5-12 above, the five cases are as follows

Case 1: $a_1 = 1$, $a_2 = 1$, $a_3 = 0$

This leads to the following equation

$$y = x_1 \oplus x_2 \qquad 5\text{-}13$$

This indicates that either of the two inputs (regulatory variables) can switch the gene on, i.e. each is an activator, but when they are both present they counteract each other and the gene remains off. This is obvious from the exclusive OR form of the Boolean function in equation 5-13. This function has algebraically appealing features as it is a linear function, i.e. it does not include products of the variables.

Case 2: $a_1 = 1$, $a_2 = 0$, $a_3 = 1$

$$y = x_1 \oplus x_2 x_1 \qquad 5\text{-}14$$

This case indicates that the presence of x_1 is necessary for the activation of the gene, since the output cannot be 1 unless x_1 is also 1. However, it is not sufficient since having $x_1 = 1$ on its own does not guarantee that $y = 1$, because if in addition we have

$x_2 = 1$, then y will be 0 in spite of x_1 being 1. On the other hand, a similar argument reveals that x_2 is sufficient on its own for repression (preventing the gene from being expressed, i.e. causing $y = 0$) because when $x_2 = 1$, we have two cases either $x_1 = 1$ which gives $y = 0$, or $x_1 = 0$ which also gives $y = 0$. However, x_2 is not necessary for repression, since it can be achieved in the absence of x_2 if x_1 is also absent.

A well known example of this case is the *lac* operon where x_1 represents lactose, as the operon cannot be turned on unless lactose is present. However, its presence does not guarantee that the operon will be on because if glucose is also present the operon will not be turned on. On the other hand, x_2 represents glucose as its presence guarantees repression, but repression can also occur without it if in addition lactose is not present. See chapter two for a detailed discussion of the *lac* operon.

Case 3: $a_1 = 0, a_2 = 1, a_3 = 1$

$$y = x_2 \oplus x_2 x_1 \qquad 5\text{-}15$$

This case is the same as case 2, except that now x_2 is the activator while x_1 is the repressor.

Case 4: $a_1 = 0, a_2 = 0, a_3 = 1$

$$y = x_2 x_1 \qquad 5\text{-}16$$

This is a synergistic case in which the gene cannot be switched on unless both inputs are present. Any one of the regulatory factors on its own is not sufficient to switch the gene on, both are necessary. This is essentially an AND gate.

Case 5: $a_1 = 1, a_2 = 1, a_3 = 1$

$$y = x_1 \oplus x_2 \oplus x_2 x_1 \qquad 5\text{-}17$$

This is equivalent to an OR gate (see equation 5-6 above) where any one of the regulatory factors on its own is sufficient to turn the gene on. When both factors are available then the gene will also be switched on. The different cases are summarised in table 5-6.

Table 5-6: Different biological explanations for the two variable Reed-Muller expansion with $a_0 = 0$. Note that the a_i's are the coefficients in the expansion while x_1 and x_2 are the inputs.

Case	a_1	a_2	a_3	Equation	Explanation
1	1	1	0	$y = x_1 \oplus x_2$	Only one or the other on its own (but not both together) can switch the gene on.
2	1	0	1	$y = x_1 \oplus x_2 x_1$	x_1 is necessary but not sufficient to turn the gene on x_2 is sufficient but not necessary to turn the gene off.
3	0	1	1	$y = x_2 \oplus x_2 x_1$	x_2 is necessary but not sufficient to turn the gene on x_1 is sufficient but not necessary to turn the gene off.
4	0	0	1	$y = x_2 x_1$	Both are necessary together to switch the gene on.
5	1	1	1	$y = x_1 \oplus x_2 \oplus x_2 x_1$	One or the other or both together can switch the gene on, (i.e. either is sufficient).

When a_0 is 1 this represents the case where the gene in normally on and the regulatory proteins either switch it off or keep it on. From the properties of the XOR operator we know that combining 1 with a variable (or a function in general) gives its complement as demonstrated by equation 5-7 above. Thus analysis similar to the case of $a_0 = 0$ can be carried out to give similar results with the appropriate interpretation, and we get the second set of five cases that are counterparts to the five above. As an example, we consider the counterpart to case 1 above, namely

Case 6: $a_1 = 1$, $a_2 = 1$, $a_3 = 0$

This leads to the following equation

$$y = 1 \oplus x_1 \oplus x_2 \qquad 5\text{-}18$$

This indicates that either of the two inputs on its own can switch the gene off, hence acting as a repressor. But when they are both present at the same time they counteract each other and the gene remains on.

The argument for a larger number of variables can be extrapolated from that for the two variable case as will be discussed in the next chapter. We have demonstrated one benefit of the RM expansion, namely that it gives a different biological insight into

the equation. However, this is not the only benefit, next we consider how when interpreted as a Boolean function with XOR operator it can facilitate the identification of the gene regulatory function.

5.5 Application to the reverse engineering of gene regulatory functions

Reverse engineering of gene regulatory function is the process by which the dependence of the expression of a gene on the different conditions affecting it is determined. The abstract form of this problem is known in the system engineering literature as system identification and is part of the modelling process of a system. The underlying idea in its basic form is that a description of the system can be inferred through exciting it with certain inputs and measuring the corresponding outputs.

System identification is essentially the design of an experiment and involves several steps, starting with the choice of an appropriate model or equation to fit the measurements to, known as the model structure problem. Another step is choosing the inputs that will excite (or in layman's terms, tease out) the different behavioural modes of the system. After applying the inputs and measuring the corresponding outputs, the parameters are computed using any of a variety of algorithms, each with its own merits and drawbacks. After that comes the problem of model validation whereby the model is tested against actual measurements and if unsatisfactory adjusted, and the process repeated until satisfactory according to preset criteria. There are many textbooks that discuss the different aspects of system identification; one of the well known ones is that by Ljung (1998).

The same underlying concepts apply when identifying Boolean functions representing gene expression regulation (D'Haeseleer *et al.* 2000; Lahdesmaki *et al.* 2003). Again there are different approaches to the problem surveyed in the literature (Camacho *et al.* 2007; Cho *et al.* 2007). Normally in reverse engineering studies the measurements represent a time course of gene expression, i.e. a time series and hence they represent a dynamic process. Recall from chapter three that dynamics in Boolean networks are

modelled by sequential circuits while static relationships are modelled by combinatorial (combinational) ones. Note that it is the same genetic network being identified in either case, and there is a debate as to which is more representative of the network, dynamic or static measurement. Dynamic measurements describe a time course but of a single biological process under a fixed set of environmental conditions and hence the measurements of the different variables will be correlated. Static measurements on the other hand describe the outputs corresponding to different input conditions and hence can be regarded as more exhaustive in coverage, (Akutsu *et al.* 1999; D'Haeseleer *et al.* 2000; Lahdesmaki *et al.* 2003). In theory, because the set of input combinations is finite, it can be applied exhaustively, however in practice this is not always possible. To identify an n variable binary function we need 2^n combinations of the binary inputs, for example when n is two we have four combinations as outlined in table 5-2 above, however this number grows exponentially with n and quickly becomes impractical. Because of that, there have been several attempts to develop methods to reduce the number of data points needed while retaining an acceptable accuracy. One such algorithm is developed by Akutsu *et al.* (1999) that significantly reduces the number of data points needed for a large network of Boolean nodes, but requires the Boolean function for each node to be limited to two inputs, hence emphasising further the special value of the two input function.

This work is concerned with combinatorial functions, and hence we will limit our discussion to reverse engineering based on measurements representing different input conditions. One study of this sort is the extensive one performed by Setty *et al.* (2003) on the *lac* operon, where different concentrations of the regulatory molecules were applied and the corresponding gene expression levels measured.

In the choice of the equation to identify (i.e. the model structure), a canonical form is desirable because it gives a unique representation for a given function, meaning that different functions will have different equations. We have demonstrated two canonical representations for Boolean functions, the DNF and the RM expansion. As is clear for the DNF from equation 5-8 above, the connective for the different terms in the equation is the OR operator, represented by the + sign. This is why the DNF is often referred to in the engineering literature as the canonical sum of products,

because it is an "ORing" of AND terms. On the other hand, in the RM expansion the connective is the XOR operator as is evident from equation 5-1, and the terms being connected are not all AND terms, some of them consist of just a single variable, i.e. not a product term.

The logic operator OR has poor discriminating ability which makes the DNF a bad choice from an identification viewpoint. On the other hand XOR has a much superior discriminating ability. The truth tables for both, together with the NAND operator are shown in table 5-7, for three inputs.

Table 5-7: Comparison of the logic operators OR, NAND and XOR.

Inputs			Output		
x_3	x_2	x_1	OR	NAND	XOR
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	1	0
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

It is clear from table 5-7 that for the OR operator the output is unchanged for seven out of the eight inputs. This is a very inefficient representation of the function since it is unable to distinguish between seven out of the eight inputs, i.e. inspecting the output corresponding to the seven inputs does not reveal any information about which of the inputs is applied. On the other hand, the XOR operation maintains a minimum Hamming distance of 2 between inputs that give the same output value. Recall that the Hamming distance between two binary numbers is the number of digits in which they differ, for example 001 and 010 are different at the first two positions from the right, while the third digit is the same, hence they have a Hamming distance of two. Inspecting table 5-7, we find that for any two outputs with the same value, the Hamming distance between the corresponding inputs is 2. For example the inputs that give an output of 1 are {001, 010, 100, 111}, and the Hamming distance between any two of them is 2; similarly with any inputs causing an output of 0. Thus XOR provides a powerful building block for representation of logic functions.

Furthermore, the outputs of the OR and NAND operators are identical for six of the eight input conditions, as evident from table 5-7. This means that six of the inputs cannot discriminate between the two functions. In general, the operators OR, NAND, AND and NOR cannot distinguish between $2^n - 1$ of the 2^n input combinations that can be applied to them, as such they are the most inefficient of all possibly binary operators (Hurst 1978).

This discussion points to a potential benefit of the RM expansion over the DNF with regard to system identification, in general leading to more efficient experiments.

As a clarification regarding table 5-7, in computing the truth values of the XOR operation the associative property is used. It is clear from the table that XOR is an odd parity operation, i.e. it gives a 1 in the output when the number of 1s in the input is odd. This applies to any number of input variables.

5.6 Combinatorial gene regulation as a polynomial on a finite field

The Reed-Muller expansion in equation 5-1 is a polynomial in two variables on the Galois field GF(2). With the appropriate analogy to real valued polynomials it can be given a Taylor series like interpretation, which is an expansion for real valued functions. So first let us review some facts about polynomials on the real numbers field and about the Taylor series expansion. As usual we will start with the simple case to be able to illustrate the concepts more clearly.

The definition of a polynomial familiar to all school students is that it is a function of the form

$$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n \qquad 5\text{-}19$$

where n is a positive integer and is the degree of the polynomial, provided that a_n is not zero. The coefficients a_i, where i = 0, 1, ..., n, and the values that the variable x can take are all real numbers, and hence the resulting values of $f(x)$ are also real numbers. More abstractly, the coefficients and the variable can in general belong to

any field including finite fields, with addition and multiplication being the operations of that field. Consequently the values of the function will also belong to the same field. In fact polynomials have a fundamental role in the construction of finite fields, and there are other definitions of a polynomial that relate to that.

Another concept that is familiar to engineering students is that of the Taylor series expansion. The idea is that a continuous and infinitely differentiable function $f(x)$ can be represented around zero by a power series given by

$$f(x) = f(0) + f_{(1)}(0)x + f_{(2)}(0)\frac{x^2}{2!} + ... + f_{(k)}(0)\frac{x^k}{k!} + ... \qquad 5\text{-}20$$

Where $f_{(k)}(0)$ is the k^{th} derivative of the function evaluated at the point $x = 0$. This expansion is also known as the MacLaurin expansion, with the term Taylor expansion indicating the general case which includes the expansion around points other than 0 as well. Of course this representation of the function is valid only for the range of values of x where the series converges. In practice only a limited number of terms is calculated and higher order terms are ignored which means that the function is approximated by a polynomial, say of degree n.

For a function of two variables x and y, the Taylor series expansion around the point (0,0) is given by

$$f(x,y) = f(0,0) + [f_x(0,0)x + f_y(0,0)y] + \frac{1}{2!}[f_{xx}(0,0)x^2 + 2f_{xy}(0,0)xy + f_{yy}(0,0)y^2] + ... \qquad 5\text{-}21$$

where f_x and f_y are the partial derivatives of the function with respect to x and y respectively, and the other subscripts indicate higher order partial derivatives in the corresponding orders.

Comparing equation 5-21 with equations 5-1 and 5-4 of the RM expansion, we notice some similarity which we will now clarify by introducing the concept of the Boolean difference.

5.6.1 The Boolean difference

The concept of the Boolean difference was introduced by Reed in his original paper of 1954 (Reed 1954) and was later developed further by him and other authors (Akers 1959; Reed 1973; Thayse and Davio 1973). It is an adaptation of the usual derivative of a function on the real numbers field to the case of the Galois field of order 2, GF(2). We will introduce the Boolean difference here in a way that we believe is much simpler and more intuitive than the way it is normally introduced in the literature, and that is by resorting to concepts the reader is already familiar with.

Recall that the derivative of a real function $f(x)$ with respect to the real variable x is the rate of change in the value of the function corresponding to the change in the value of the variable, given by

$$\lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \qquad 5\text{-}22$$

Now on GF(2) the variable x can only take the values 0 and 1, hence the only change in x will be by the value of 1 giving $\Delta x = 1$. In this case the limit has no meaning and we get the Boolean difference rather than derivative. Note further that on GF(2) addition and subtraction are the same since 1 is its own additive inverse. Now applying those facts to equation 5-22 and noting that we are using modulo 2 addition, we get

$$\Delta f(x) = f(x \oplus 1) \oplus f(x) \qquad 5\text{-}23$$

Using equation 5-7 above, this can be re-written in the form

$$\Delta f(x) = f(\bar{x}) \oplus f(x) \qquad 5\text{-}24$$

This is an intuitive result since on GF(2), the variable x can only take two values namely a given value and its complement. Hence the Boolean difference of a function with respect to its variable is simply given by

$$\Delta f(x) = f(0) \oplus f(1) \qquad 5\text{-}25$$

It should be noted that, unlike derivatives with respect to real variables, the Boolean difference does not distinguish the direction of change in the function. So a change in

the function value from 0 to 1 cannot be distinguished from a change from 1 to 0, this is because as mentioned earlier, on GF(2) 1 and -1 are the same.

In the case of more than one variable, like in the real valued case were we get partial derivatives, here we get partial differences with respect to the relevant variables. So the Boolean difference of the function of two variables $f(x_2, x_1)$ with respect to x_1 is obtained by changing x_1 but not x_2, this is given by

$$\Delta_{x_1} f(x_2, x_1) = f(x_2,0) \oplus f(x_2,1) \qquad 5\text{-}26$$

Note that this function does not depend on x_1 anymore and is only a function of x_2 which means that the second difference of $f(x_2, x_1)$ with respect to x_1 will be zero. It also means that the second difference with respect to x_2 (of the first difference with respect to x_1) will now be given by

$$\Delta_{x_2 x_1} f(x_2, x_1) = [\Delta_{x_1} f(x_2 = 0)] \oplus [\Delta_{x_1} f(x_2 = 1)] \qquad 5\text{-}27$$

Using equation 5-26 this becomes

$$\Delta_{x_2 x_1} f(x_2, x_1) = [f(0,0) \oplus f(0,1)] \oplus [f(1,0) \oplus f(1,1)] \qquad 5\text{-}28$$

Note that this function does not depend on x_2 nor on x_1. Thus any higher order differences will be zero. Now let us look at those differences evaluated at the point (0,0), we get from equations 5-26 and 5-28 the following

$$\begin{aligned} \Delta_{x_1} f(0,0) &= f(0,0) \oplus f(0,1) \\ \Delta_{x_2} f(0,0) &= f(0,0) \oplus f(1,0) \\ \Delta_{x_2 x_1} f(0,0) &= f(0,0) \oplus f(0,1) \oplus f(1,0) \oplus f(1,1) \end{aligned} \qquad 5\text{-}29$$

Those together with the value of $f(0,0)$ are exactly the same as equations 5-4 above of the coefficients of the Reed-Muller expansion. Hence the Reed-Muller expansion can be written as

$$f(x_2, x_1) = f(0,0) \oplus \Delta_{x_1} f(0,0) x_1 \oplus \Delta_{x_2} f(0,0) x_2 \oplus \Delta_{x_2 x_1} f(0,0) x_2 x_1 \qquad 5\text{-}30$$

This has exactly the same form of the Taylor series expansion around the point (0,0) given by equation 5-21, noting that second order differences with respect to the same

variable and all higher order differences with respect to all variables are zero. Hence the RM expansion can be viewed as a Taylor series expansion on GF(2) taking into account the difference in the interpretation of course of some of the mathematical concepts involved. Again, strictly speaking, equation 5-30 is the MacLauren series expansion. One however, can expand the function around other points to get a Taylor series expansion (Akers 1959; Thayse 1974b), although in such a case the coefficients will not correspond to those of the RM expansion. Furthermore, the Taylor series expansion whether around the point (0,0) or any other point indicates that one can construct the function (i.e. identify the parameters or reverse engineer it) solely by knowing its Boolean differences around some point, without the need for knowing the actual values of the function around a set of different points (Akers 1959). This can be verified from equation 5-30 by using a substitution like that used for equations 5-2 above.

From a biological perspective, this Taylor series interpretation means that the coefficients of the RM expansion provide sensitivity information of the gene regulatory function with respect to the different regulating variables. More interestingly however, the Boolean difference can be used as a tool to detect mutations in the DNA that affect the function of a gene. This task can also be performed using the RM expansion without resorting to the Boolean difference because formulation of a logic function in the form of the RM expansion allows it to be easily tested for certain types of faults (Reddy 1972). By placing the problem of detecting mutations as a problem of detecting faults in a logic circuit we can utilise the methods used for the latter to detect such mutations. To do that we first need to introduce fault detection in logic circuits, then introduce mutations and then establish the correspondence between them that would allow us to apply the former to the latter.

5.6.2 Fault detection in logic circuits

Fault detection is a growing area within the field of logic design because of the increasing complexity of electronic circuits. We will only present here the concepts that will help us migrate some of the techniques of fault detection to the problem of mutation detection. No attempt is made here to give an even concise survey of the

field, however, certain fundamental concepts need to be introduced irrespective of what we will cover.

There are three related concepts underlying fault diagnosis, those are failure, error and fault. A failure is said to occur when the circuit, or a system in general, does not perform the function it is designed to do. A failure is caused by an error in the operation of the circuit or some subpart of it, which means that it will be in a state other than the one it should be in. An error is caused by a fault which is a physical damage of some sort, such as a short circuit. Those and other concepts are covered in the comprehensive text by Jha and Gupta (2003) from which we take the following example to clarify them. Consider a motor car; a puncture in a tyre is a physical damage resulting in the tyre being deflated, i.e. its pressure having a different value (is in a different state) from the correct one, and hence the car cannot travel. By analogy to the concepts above, the puncture is the fault, the low pressure in the tyre is the error and the inability to travel is the failure.

Note that whilst a failure is caused by an error which in turn is caused by a fault, the converse is not necessarily true. In other words, not all faults cause errors and not all errors cause failure. This is the principle behind fault tolerant design whose purpose is to ensure that the function will not be interrupted when a fault occurs, which is crucial in applications where maintenance is very difficult, costly or dangerous. Such situations occur for example in the case of a space craft in outer space, or a pacemaker inside a patient's body.

Other concepts related to faults which we will later translate to the context of mutation are the cause and effect of faults. In electronic circuits, a fault can be caused by problems in manufacturing thus producing a defective circuit, or it can occur during operation such as damage due to applying the wrong voltage, environmental conditions such as excessive temperature or radiation, or aging of components. Normally the ultimate effect of a fault is failure.

Fault diagnosis involves two processes namely fault detection and fault location, i.e. locating the point in the circuit where the fault occurred. To be able to detect a fault, it must cause an error, otherwise it will not be detectable. Furthermore, faults must be

distinguishable from each other otherwise we might detect the occurrence of an error but cannot find the fault causing it. The obvious approach to finding a fault is to apply all the possible combinations of inputs to the circuit, measure the corresponding outputs and compare the results with those of the fault-free circuit to decide on the fault. As with the case of system identification discussed above, this approach is not practical as it is expensive and time consuming not to mention that applying all inputs to a faulty circuit can cause additional problems and faults. Thus there is a need for a more rational approach to the problem other than the brute force one, whereby a formal process can be followed in a systematic way that will produce a minimum number of tests (Lala 1997; Jha and Gupta 2003). There are several such methods and they are based on modelling both the errors and the faults.

As is often the case, models can be formulated at different levels of abstraction, as was mentioned in chapter three in the context of modelling gene regulatory networks. In the case of modelling circuits for the purpose of error and fault detection, the levels of abstraction include investigating at the system level, subsystems, gate level (such as AND, OR and other gates) or the device level, where devices such as transistors are used to implement gates (depending on the technology). Error models are mainly probabilistic models concerned with the probability of occurrence of the different errors and the correlation between them. Fault models are representations of the possible physical problems that can happen on a circuit and include so called stuck-at faults and bridging faults among others.

The stuck-at fault model which we will use in our analysis means that the logic value at some point or line in the circuit is stuck at some value and does not change irrespective of the change of other signals in the circuit affecting that point. For a binary circuit the point can be either stuck at 0 or stuck at 1, denoted by s-a-0 and s-a-1 respectively (Jha and Gupta 2003).

We will consider two methods for fault detection, the first which we will derive and discuss in some detail, is based on the Boolean difference. The second method uses the RM expansion, and we will only describe the principle behind it without the details. Both methods have the same underlying principle namely applying a carefully chosen set of inputs to the circuit, observing the output, and in most cases, comparing

it with the error free output. As mentioned above, for this approach to work the inputs applied have to excite the error in the presence of the fault. For example if one of the two inputs to an AND gate is stuck at 0, then of the four possible input combinations that can be applied, namely (0,0), (0,1), (1,0) and (1,1), only the last one will cause an error at its output, see table 5-4. Furthermore, it should be possible to detect the error at the output. This is important because internal points in the circuits may not be accessible, such as in the case of packaged integrated circuits.

Those two concepts, namely exciting the error and detecting it at the output, are known as error generation and error propagation respectively. The latter means that if the fault causes an error at an internal point in the circuit, the test inputs should be chosen to guarantee the propagation of this error to the output. It is worthwhile to note that the concepts of error generation and propagation relate to similar concepts in control theory known as controllability and observability. Controllability investigates whether the system can be forced into a given state by an input, while observability is concerned with whether a given state can be detected from the output, both having impact on the testability of a system.

To derive the method for fault detection, assume that if a circuit is fault free its output will be $f(x)$, and if it has a fault then it will implement a different function, call it $f_e(x)$ for function with error. To be able to detect the fault, the two functions must be different. So for a two variable function for example, we must have

$$f(x_2, x_1) \neq f_e(x_2, x_1) \qquad 5\text{-}31$$

Since on GF(2) any value is its own inverse, then by adding $f_e(x_2, x_1)$ to both sides we get

$$f(x_2, x_1) \oplus f_e(x_2, x_1) \neq 0 \qquad 5\text{-}32$$

And since on GF(2) the only other value than 0 is 1, we get the condition for detecting a fault in a logic circuit as

$$f(x_2, x_1) \oplus f_e(x_2, x_1) = 1 \qquad 5\text{-}33$$

This equation tells us the condition for the existence of an error, but this error might be caused by one or more faults. Since our purpose here is only to demonstrate the method, we will use the simplest fault model, and that is a single fault of the stuck-at type and that it occurs at one of the inputs. Let us assume that the fault occurs at the input x_1 then to detect a stuck-at 0 fault we apply a value of 1 at the x_1 input. Now due to the fault, the value of 1 will be seen by the circuit as a value of 0 and hence it will produce the erroneous output $f(x_2, 0)$ instead of the correct output $f(x_2, 1)$. Similarly for a stuck-at 1 fault we apply the value of 0 at the corresponding input. Thus to detect a stuck-at value fault on one of the inputs, we apply the complement of the value at that input to invoke an erroneous output. In this case for a stuck-at fault at the x_1 input, equation 5-33 becomes

$$f(x_2, x_1) \oplus f(x_2, \bar{x}_1) = 1 \qquad 5\text{-}34$$

On comparison with equation 5-26 above, we find that the left hand side of equation 5-34 is the Boolean difference of the function with respect to x_1 given by

$$\Delta_{x_1} f(x_2, x_1) = f(x_2, x_1) \oplus f(x_2, \bar{x}_1) \qquad 5\text{-}35$$

Hence the condition for detecting a single stuck-at fault at one of the inputs of a logic circuit is that the Boolean difference of the function implemented by the circuit, with respect to that input is 1, i.e.

$$\Delta_{x_1} f(x_2, x_1) = 1 \qquad 5\text{-}36$$

The solution of this equation on GF(2) gives the values of the inputs that will guarantee detection of the fault, i.e. its generation and propagation. Reed (1973) gives the condition for the existence of a solution to this equation. In fact he applies it to the more general case of multiple faults and does not limit his analysis to faults at the inputs (Sellers *et al.* 1968; Reed 1973). In general there can be more than one set of values for the variables that satisfies equation 5-36, each set is known as a test vector. For a function of many variables this procedure can be repeated for each input and will give a number of test vectors for each input, some of which may overlap.

The second method for fault detection uses the Reed-Muller expansion directly without the need for resorting to the Boolean difference. As is clear from equation 5-1

above, the RM expansion is a modulo 2 sum of modulo 2 product terms. The modulo 2 sum is an odd parity operation as discussed above where the XOR operation implements the modulo 2 addition. Hence the idea behind fault detection using the RM expansion is to choose the appropriate pattern of inputs that will give the appropriate number of 1s from the product terms that will be propagated to the output. This makes it very easy to test for internal faults, in fact an all 0s input and all 1s input can detect a fault on any of the modulo 2 adders irrespective of the function implemented. Similarly it is very easy to test for faults at the multipliers inputs and outputs. Details of the development of the method and examples of its application can be found in the literature (Reddy 1972; Akers 1987; Damarla and Karpovsky 1989; Gil and Ortega 1998).

5.7 Application to the detection of mutation

We now turn our attention to the main reason for introducing the material on fault detection, and that is to apply it to the detection of mutations in a gene. We will map the concepts introduced above in the context of electronic engineering to genetics, starting with the cause and effect of mutations.

Recall that a mutation is a change in the DNA sequence of the cell. Generally speaking there are two main causes for such change. The first is a spontaneous change whereby a nucleotide is erroneously copied during DNA replication which is part of the cell division process; and is known as a replication error. The occurrence of such an error is very rare and its rate is normally around one base in every 10^{10}, i.e. one base in every ten billion is miscopied. Whilst replication errors are internally caused, change in the DNA sequence can also be caused by external factors, collectively known as mutagens, i.e. mutating agents or factors causing mutation. This is the second possible cause of a mutation and is an induced mutation, as opposed to the spontaneous one caused by replication errors. Mutagens can be broadly classified into chemical, physical and biological factors (figure 5-1), all three cause either damage or alterations to the DNA. Chemical factors such as carcinogens react with the DNA and modify it, while physical factors such as radiation can damage the DNA. Biological factors such as retroviruses integrate its own DNA into that of the host organism hence altering its genetic makeup (Winter *et al.* 2002).

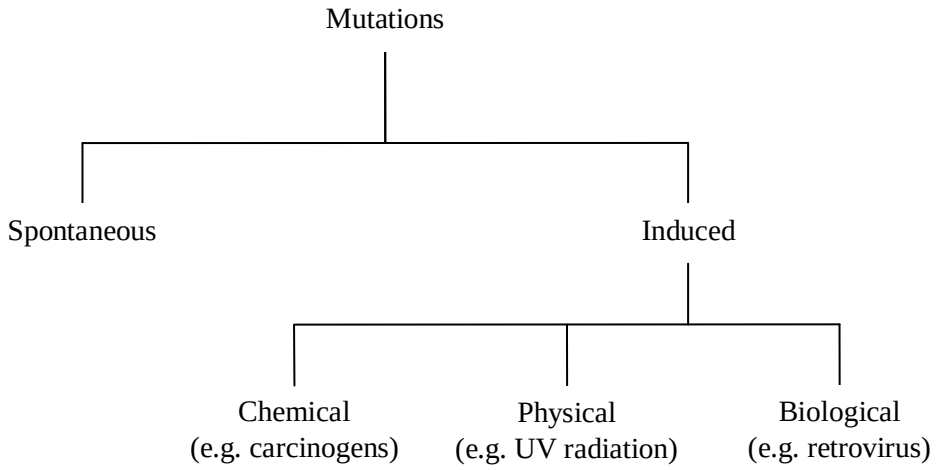


Figure 5-1: Causes of mutations.

On an abstract level one can note a correspondence between the causes of mutations in a cell and the causes of faults in a circuit. Spontaneous mutations can be regarded as corresponding to faults occurring during manufacturing of a circuit, in the sense that they are both less common and occur before the fact. In other words, spontaneous mutations occur before the cell even starts its life and gets exposed to the different environmental factors. Correspondingly manufacturing defects occur before the circuit is even put into operation and gets exposed to adverse operating and environmental conditions. On the other hand, induced mutations correspond to faults that occur during operation of a circuit in the sense that they are caused by external factors. For a cell those are factors from the environment in which the cell lives, or correspondingly for a circuit the conditions in which the circuit is operated.

The effects of mutations will depend on their type. Mutations are broadly classified into point mutations and gross mutations. Gross mutations involve the alteration of a large chunk of DNA such as deletion or swapping parts of a chromosome with each other, and they can cause major problems in the organism. Point mutations on the other hand involve the change of a single nucleotide (Winter *et al.* 2002). In analogy with fault detection, we can think of point and gross mutations as single and multiple faults respectively. Here we will focus on the different types of point mutations and their effects.

The effect a point mutation will have on the coded protein will depend on the nature of the mutation, and there are four major types. Recall from chapter two that a protein consists of a string of amino acids, each coded by a three nucleotide (or base) codon. Each nucleotide in the DNA can be one of four types A or G (both purines) or T or C (both pyrimidines), that correspond to A, G, U or C in the RNA. Let us examine the four main types of point mutations by considering how the change in any of the three nucleotides of a codon affects the resulting amino acid. We will demonstrate this using the amino acid Leucine (table 5-8).

Table 5-8: Examples of mutations in the first, second and third base of the codon for the amino acid Leucine.

1st base	2nd base	3rd base	Amino acid	Side chain	Remark
U	U	A	Leucine	Hydrophobic - Aliphatic	
U	U	G	Leucine		Same amino acid
U	A	A	STOP codon		
A	U	A	Isoleucine	Hydrophobic - Aliphatic	Same protein

The code for Leucine is UUA, when the third base (A) changes into G, we still get the same amino acid, hence there is no change in the resulting amino acid. This is known as a silent mutation. When the second base changes from U to A, we get a STOP codon instead of an amino acid, and is known as a nonsense mutation. When the first codon changes from U to A, we get the amino acid Isoluecine, and is known as a missense mutation (figure 5-2).

Silent mutations do not cause any change in the amino acid, they normally result in a synonym of the original amino acid, i.e. a different code for the same amino acid. This is normally the case when the mutation is in the third base of the codon. Missense mutations result in a different amino acid, which if of similar characteristics to the original one will not normally affect the resulting protein, as with the example of Leucine and Isoleucine above, and is known as a neutral mutation (Russell 2006). Some missense mutations however, may result in a major alteration of the conformation and hence the function of the resulting protein. A well known example is the mutation of glutamic acid to valine in the protein beta-globin, resulting in sickle

cell anaemia. Nonsense mutations result in the early termination of the translation of a protein, hence an incomplete protein that will not perform the intended function, potentially resulting in a major problem and often a different phenotype.

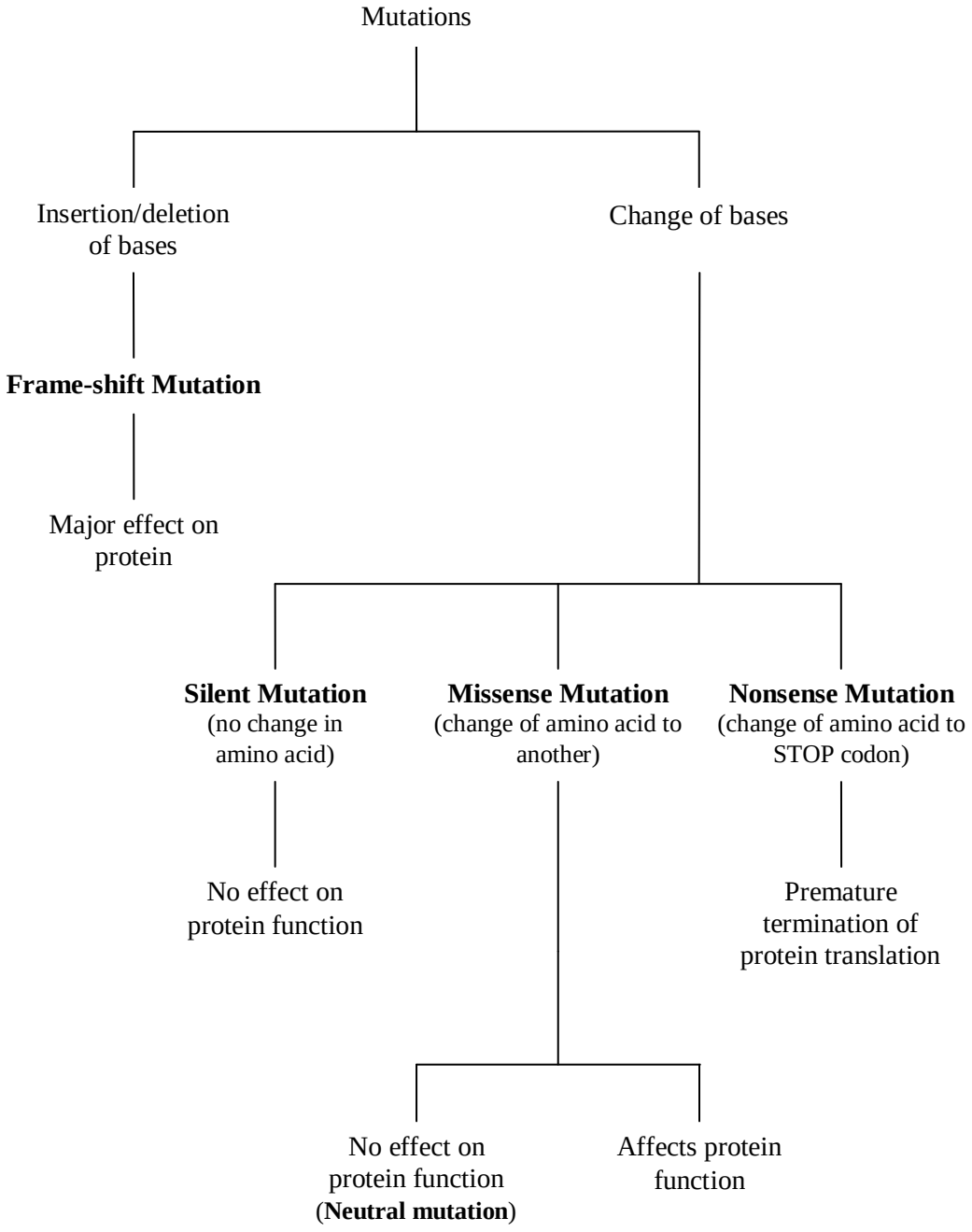


Figure 5-2: Types of mutations and their effects on protein function.

The fourth type of mutation is called a frame shift mutation, and results from inserting or deleting a nucleotide in the sequence of a gene; since each codon

comprises three nucleotides this will cause an error in the resulting sequence of amino acids, again leading to a mutant phenotype (Winter *et al.* 2002).

Given the different types of mutations, how can one detect that a mutation actually occurred? The common technique is to sequence the genome and compare it with that of the "wild type". Here however, we are interested in the change in function (if any) resulting from the mutation, rather than the details of the nucleotide alterations. It should be noted that when sequencing a genome for the first time, the functions of some genes might not be known, especially in eukaryotes. In such a case the sequence of a gene is compared with that of a similar one in another organism whose function is known, hence one can assume the function of the gene in question. This process is known as homology analysis and relies heavily on computational and bioinformatics tools, but it is not always accurate in eukaryotes.

In analogy with faults in electronic circuits, and since we are interested in the function performed by a protein rather than its composition, then any mutation that does not affect this function will not be detected. Thus silent mutations are analogous to faults that do not cause an error. Neutral missense mutations that do not affect the function of the resulting protein are analogous to faults that cause an unobservable error. Finally any mutation that affects the function of the resulting protein such as some missense mutations, most nonsense and frame shift mutations are considered as errors causing failure.

Let us now consider a simple example to demonstrate these concepts and tools. Mayo *et al.* (2006) have preformed a detailed study on the *lac* operon where they made several mutations to different sites on the DNA molecule to test how this affects the robustness (referred to as plasticity) of the gene regulatory function against such mutations. The *lac* operon was explained in some detail in chapter two; recall that it is controlled by two regulatory proteins, Lac repressor and Catabolite Activator Protein (CAP). Mayo *et al.* (2006) made mutations to the sites on the DNA molecule where those two proteins bind, hence altering their effect on the regulatory function. They made several point mutations to each site and in some experiments to both sites simultaneously. Since we are interested in the effect of the mutations on the function, the number of point mutations on a given site will not affect our study as long as it is

on one site at a time. In their extensive study they obtained up to twelve different logic functions, depending on the number and location of the mutations. Recall from chapter two that the operator site consists of three non-contiguous regions which explains the large number of functions. We will use the results of one of their experiments which included mutations to the binding site of Lac repressor but not CAP. Table 5-9 below shows the results of both the un-mutated (wild type) case and the mutant. In line with our notation above we have called them $f(x_2, x_1)$ and $f_e(x_2, x_1)$ respectively.

Table 5-9: The *lac* operon regulatory function for the wild type and a mutant.

Glucose x_2	**Lactose** x_1	**Wild type** $f(x_2, x_1)$	**Mutant** $f_e(x_2, x_1)$
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

This example will allow us to demonstrate two concepts, firstly how to use the RM expansion to model a gene regulatory function, and secondly how to detect mutations using the methods above. First let us consider the modelling task. Table 5-9 contains the truth vectors for the two regulatory functions; substituting those in equation 5-5 above we get the coefficients for the polynomial of the RM expansion. Substituting the coefficients in equation 5-1, we get the two functions below.

$$f(x_2, x_1) = x_1 \oplus x_2 x_1 \qquad 5\text{-}37$$

$$f_e(x_2, x_1) = x_2 x_1 \qquad 5\text{-}38$$

We will assume that we do not know which of the two factors the mutation affects. So we assume that we performed an experiment where for each of the two factors we applied the two limits, namely none and maximum as the authors have done, and obtained the results of $f_e(x_2, x_1)$ in table 5-9. Now substituting the two functions of equations 5-37 and 5-38 into equation 5-33 above, we get the condition of the existence of a mutation is

$$[x_1 \oplus x_2 x_1] \oplus [x_2 x_1] = 1 \qquad 5\text{-}39$$

Noting that the addition is on GF(2), the condition now becomes

$$x_1 = 1 \qquad 5\text{-}40$$

This indicates that the error occurs only when x_1 = 1, i.e. when lactose is applied. Indeed this can be verified from table 5-9 by inspection of both cases, the wild type and the mutant, where it is clear that the error occurs only when x_1 = 1 indicating that there has been a mutation in the part responsible for lactose.

Now knowing that the fault is in x_1, in order to find the test vector, i.e. the values of the inputs that will let us observe the error in the output, we take Boolean difference of the original function with respect to the faulty input x_1. So using $f(x_2, 0)$ and $f(x_2, 1)$ from equation 5-37 and substituting in equation 5-35 we get

$$\Delta_{x_1} f(x_2, x_1) = x_2 \qquad 5\text{-}41$$

Using equation 5-36 we get

$$x_2 = 1 \qquad 5\text{-}42$$

which means that the error will only be observed when x_2 = 1, i.e. when glucose is applied, which is evident from table 5-9.

Hence equation 5-40 specifies the conditions for error generation, while equation 5-42 specifies the conditions for error propagation. This is satisfactory from the point of view of fault detection, however, from a biological viewpoint it does not tell us much about the nature of the mutation. Again in fault detection we can conclude that this is a stuck at zero fault because it only happens when x_1 = 1, implying that the system sees the 1 as a 0, hence stuck at zero. However, the biology is more complicated, we know that the mutation is related to the lactose processing but we are not sure what it is exactly. The biological equivalent of stuck at zero is that the cell does not effect the action of lactose. This can be due to a problem with Lac repressor, either the protein itself or the expression of the gene *LacI* that codes it; it can also be in its binding site on the DNA molecule. It so happens that in this particular case we know that the problem is in the binding site, but the issue to note here is that whilst mathematics gives us the result we need biology to interpret it. This relates to the earlier discussion

in chapter three about the need for the domain specific knowledge to interpret the mathematical results, (see figure 3-2 in chapter three).

5.8 Summary and Conclusion

In this chapter we have presented a simple development of the Reed-Muller expansion of a logic function. It differs from the disjunctive normal form (DNF) commonly used in the analysis and design of logic circuit in that it considers the function on the finite field GF(2) rather than on a Boolean algebra. We then gave the RM expansion two different interpretations on two algebraic structures. For each algebraic interpretation we demonstrated biological insight and functionality. Firstly we viewed the RM expansion as a function on Boolean algebra (a ring is actually sufficient) that uses AND-XOR rather than the AND-OR-NOT operations. We have demonstrated the superiority of the discriminating power of the XOR operation compared to OR, and hence its potential value in the reverse engineering of genetic networks. We then viewed the RM expansion as a polynomial on the field GF(2) and presented a simple development of the Boolean difference which is used in fault detection in logic circuits. Hence, we suggested that when a mutation is viewed as a logic fault in the combinatorial gene regulatory function, the Boolean difference can be used for mutation detection.

This emphasises an important notion in this chapter and indeed in the whole of this report, and that is the power of abstraction. Namely that when one detaches the underlying concepts from the implementation details, one can glide the methods across the boundaries of the disciplines and possibly use methods that have been known in one field for decades but not known in the other, even though they are tools for investigating the same problems but in different contexts.

Regarding the RM expansion as a polynomial on the field GF(2), and using the Boolean difference, we drew an analogy with the Taylor series expansion which is also a polynomial on a field. However, the Taylor series expansion can also be regarded as an expansion on a function space, where now for the Reed-Muller expansion the functions are binary, presenting another analogy between the two that we will explore in the next chapter.

Chapter 6: A Transform Approach to Modelling Combinatorial Gene Regulatory Functions

6.1 Introduction

This chapter constitutes the second part of the main contribution of this work. The contribution is the theoretical development of a method for modelling discrete gene regulatory functions. In the previous chapter we have introduced the core of this development which is the Reed-Muller expansion for the binary case. We gave it two algebraic interpretations namely as a function on a Boolean algebra and as a polynomial on a finite field, each with a biological meaning and potential use. The polynomial mentioned in the second interpretation can be viewed as a Taylor series type of expansion, which as was discussed in chapter four can also be viewed as an expansion on a function space. We will start this chapter by picking up this thread from the last chapter and developing it further, in particular representing this expansion as a transform on the function space as is common in functional analysis. As with the other two interpretations in the previous chapter we also suggest an application for the transform method, namely in the emerging interdisciplinary field of synthetic biology. The second part of the development mentioned above builds on the core which is the binary case and extends it to the multiple-valued case. In analogy with the binary case we will give the development and possible biological interpretations. Unlike the binary case however, we will not go into a detailed mathematical argument but will only mention the results. We will give more emphasis to the transform form of the multiple-valued case, in particular its conceptual interpretation.

6.2 Combinatorial gene regulation as a linear transformation on a function space

We have shown in the previous chapter that the Reed-Muller expansion can be viewed as a Taylor series type of expansion on the field GF(2). We have also seen in chapter four in the context of vector spaces that the Taylor series expansion for a real function can be regarded as a linear expansion on a function space, where the individual functions are given by the different powers of the independent variable *x*. The Taylor series for a real function, repeated here for convenience from the previous chapter (equation 5-20) where the notation has been explained, is given by

$$f(x) = f(0) + f_{(1)}(0)x + f_{(2)}(0)\frac{x^2}{2!} + ... + f_{(k)}(0)\frac{x^k}{k!} + ...$$

This can be written as

$$f(x) = \sum_{k=0}^{\infty} a_k x^k \qquad 6\text{-}1$$

Alternatively it can be written as

$$f(x) = \sum_{k=0}^{\infty} a_k g_k \qquad 6\text{-}2$$

where g_k is the k^{th} power of x

For more than one variable, say n variables, this can be written in the general form

$$f(x_n,..., x_2, x_1) = \sum_{k=0}^{\infty} a_k g_k \qquad 6\text{-}3$$

where now g_k is a monomial, i.e. a single product term of the variables given by

$$g_k(x_n,..., x_2, x_1) = x_n^{k_n} ... x_2^{k_2} x_1^{k_1} \qquad 6\text{-}4$$

and the k_i's in the exponent are positive integers. Now the Taylor series becomes a linear combination of the functions g_k's rather than a polynomial in x. An example with two variables x and y was given in the previous chapter (equation 5-21). With more than two variables, the order of the variables in the equation becomes an issue.

For example for three variables x, y and z, how can we order the two monomials x^2yz or xy^2z, given that both have an overall degree of four? The degree of a monomial is the sum of the powers of its constituents, so both terms have the same degree. This issue is resolved by choosing a particular ordering of the variables in the equation and maintaining it throughout the analysis as shall be demonstrated later (Cox *et al.* 2007).

One should be careful not to confuse these two somewhat subtle aspects of the Taylor series, namely that it is both a polynomial and a linear function at the same time. The interpretation depends on what the linearity or otherwise is with respect to. So whilst equation 6-1 represents a polynomial in x or more generally in several variables, equation 6-3 represents a linear combination of functions (the g_k's). Hence the Taylor series as expressed in 6-3 is a linear combination of non-linear functions given by equation 6-4. Recall from chapter four that those functions are linearly independent, and thus they form a basis that spans the whole space of continuous and infinitely differentiable functions. Hence any such function can be represented by this linear combination, with the appropriate coefficients for the different terms.

We now consider how the Reed-Muller expansion can be viewed as an expansion on a function space. We start with the two variable case, which we will later generalise to several variables. We have introduced the RM expansion in the previous chapter as

$$f(x_2, x_1) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_2 x_1 \qquad 6\text{-}5$$

The coefficients of the RM expansion relate to the truth values of the function by the matrix equation (from the previous chapter)

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \qquad 6\text{-}6$$

which can be rewritten in compact notation as

$$\boldsymbol{a} = \boldsymbol{T}\boldsymbol{d} \qquad 6\text{-}7$$

Where $\boldsymbol{a}$ is the vector of the RM expansion coefficients given by

$$\boldsymbol{a}' = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \end{bmatrix}$$

(The prime on $\boldsymbol{a}$ indicates the transpose of the vector, this is normally denoted by the letter T but we chose to use the apostrophe in this case to avoid confusion with the matrix $\boldsymbol{T}$.)

and $\boldsymbol{d}$ is the vector of truth values of the function

$$\boldsymbol{d}' = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 \end{bmatrix}$$

and $\boldsymbol{T}$ is a transformation matrix given by

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad 6\text{-}8$$

It is well known from linear algebra that a linear transformation between two vector spaces can be represented by a matrix and conversely a matrix can represent a linear transformation between two vector spaces (Naylor and Sell 1982). Indeed the matrix $\boldsymbol{T}$ transforms the truth values to the coefficients of the RM expansion. Furthermore, from the previous chapter (equation 5-3), we have found that the equation that computes $\boldsymbol{d}$ from $\boldsymbol{a}$ has the same form as equation 6-6 above, i.e. we have

$$\boldsymbol{d} = \boldsymbol{T}\boldsymbol{a} \qquad 6\text{-}9$$

But from equation 6-7 and noting that the matrix $\boldsymbol{T}$ is invertible on GF(2), we have

$$\boldsymbol{d} = \boldsymbol{T}^{-1}\boldsymbol{a} \qquad 6\text{-}10$$

Hence we get

$$\boldsymbol{T} = \boldsymbol{T}^{-1} \qquad 6\text{-}11$$

This means that the Reed-Muller expansion can be viewed as a transformation on a vector space. In fact it is a transformation on a function space where the functions are binary valued as opposed to the real valued functions of the Taylor series expansion

in equation 6-1 above. Furthermore, this transformation is not only invertible but is its own inverse as well.

We can demonstrate that the RM expansion is a linear combination of functions on a vector space by examining table 6-1 below with all the possible two variable functions on GF(2), repeated from the previous chapter. We note from the table that any of the sixteen binary functions can be represented by a linear combination (using modulo 2 addition) of the functions f_{15}, f_{10}, f_{12} and f_8. Each of these four functions corresponds to a term in the RM expansion of equation 6-5. Furthermore, those functions are linearly independent, i.e. none of them can be represented as a combination of the other three or of any other functions in the table. Hence the functions f_{15}, f_{10}, f_{12} and f_8 span the space of all two variable binary functions, and form a basis for this space. Indeed, comparing the truth values of the functions f_{15}, f_{10}, f_{12} and f_8 with the columns of the matrix in equation 6-8 reveals that they are the same. This is not a surprise since the post multiplication of a matrix by a vector, which is the operation in equation 6-6, leads to a linear combination of the columns of the matrix with the coefficients being the corresponding elements of the vector (Strang 1988). It is worth noting from table 6-1 that the DNF of a function is a logic OR combination of the functions f_1, f_2, f_4 and f_8, which represent the min terms as explained in chapters three and five. They also form a basis (under logic OR) for the space of all two variable Boolean functions, however, working with matrices on a Boolean algebra is not as straightforward as on a field because of the unusual properties of the Boolean algebra, see chapter four for more details.

Table 6-1: All possible binary functions of two binary variables.

m	Inputs		Outputs															
	x_2	x_1	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

So why is considering the RM expansion as a linear transformation on a vector space important? We will see shortly that this makes extending the RM expansion to several variables much easier, and also gives an intuitive interpretation to synthesising logic functions. Biologically and hence more relevant to our work, this will have corresponding significance in the context of synthetic biology.

Let us now apply these concepts to the Reed-Muller expansion, where for an n variable function it is given by a form similar to that of the Taylor series expansion, equations 6-3 and 6-4 above.

$$f(x_n,..., x_2, x_1) = \sum_{k=0}^{2^n-1} a_k x_n^{b_{kn}} ... x_2^{b_{k2}} x_1^{b_{k1}} \qquad 6\text{-}12$$

Note that unlike real functions, the second and higher order differences of a Boolean function with respect to the same variable is zero and hence we end up with a finite sum as opposed to the infinite series for the continuous case.

We will choose a particular ordering of the variable where $b_{kn}....b_{k2}b_{k1}$ is the n digit binary representation of the decimal number k whose values range from 0 to 2^n-1. This is similar to the ordering of the functions above as explained in chapter five, table 5-3. For example, for a four variable function, k will range from 0 to $(2^4 - 1) = 15$ and will be represented by a four digit binary number. So for $k = 7$, its binary representation is 0111 and the corresponding term in equation 6-12 will be given by

$$a_7 x_4^0 x_3^1 x_2^1 x_1^1 = a_7 x_3 x_2 x_1$$

For three variables, equation 6-12 becomes

$$f(x_3, x_2, x_1) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_2 x_1 \oplus a_4 x_3 \oplus a_5 x_3 x_1 \oplus a_6 x_3 x_2 \oplus a_7 x_3 x_2 x_1 \qquad 6\text{-}13$$

Using the same approach as for the two variable case in the previous chapter, we get the corresponding equation for three variables, from which we can compute the coefficients a_k

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} \qquad 6\text{-}14$$

Where the monomials given by equation 6-12 represent the basis functions that span the space of all three variable Boolean functions, and k in the equation ranges from 0 to 7 (= 2^3 - 1). We will denote the basis functions by r_k, given by

$$\begin{aligned} r_0 &= 1 \\ r_1 &= x_1 \\ r_2 &= x_2 \\ r_3 &= x_2 x_1 \\ r_4 &= x_3 \\ r_5 &= x_3 x_1 \\ r_6 &= x_3 x_2 \\ r_7 &= x_3 x_2 x_1 \end{aligned} \qquad 6\text{-}15$$

Considering the r_k's as vectors, equation 6-14 can be written as

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{r_0} & \boldsymbol{r_1} & \boldsymbol{r_2} & \boldsymbol{r_3} & \boldsymbol{r_4} & \boldsymbol{r_5} & \boldsymbol{r_6} & \boldsymbol{r_7} \end{bmatrix} \boldsymbol{d} \qquad 6\text{-}16$$

where each $\boldsymbol{r_k}$ is a function representing the corresponding column vector in the matrix of equation 6-14, and they can be plotted against the decimal equivalent of the binary values of the inputs as illustrated in figure 6-1. These eight (8 = 2^3) basis functions can be linearly combined to generate any of the 256 (= 2^8) possible binary functions of three variables.

Note that whilst the Reed-Muller functions are linearly independent, they are not orthogonal. This can be verified by inspection of figure 6-1 for the three variable case, where the inner product of any two of the functions is not zero. This is true for any number of variables. See chapter four for a discussion of orthogonality.

As an aside and from a purely algebraic point of view, it is to be noted that the set of basis functions is closed under component-wise multiplication, defined as the multiplication of the corresponding components of any two columns, recall that we are using modulo 2 multiplication. This closure means that the product of any two (or more) functions in this set will also be in the set. Furthermore, $\boldsymbol{r_0}$ acts as an identity (unity) for this operation i.e. when multiplied by any of the functions it does not change it. This multiplication is also associative and commutative, hence the set forms an algebraic structure known as a commutative monoid (see chapter four). In

fact the four functions $\boldsymbol{r_0}$, $\boldsymbol{r_1}$, $\boldsymbol{r_2}$ and $\boldsymbol{r_4}$ generate the other functions in the set, for example $\boldsymbol{r_3}$ is obtained from the product of $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ (where 3 = 1 + 2), as can be verified by inspection from figure 6-1 or from equation 6-14. Similarly, $\boldsymbol{r_5}$ is the product of $\boldsymbol{r_1}$ and $\boldsymbol{r_4}$ (where 5 =1 + 4), and so on. Note that only the functions with index 2^k where k is from 0 to n-1 generate the other functions, so in the case of n = 3, those are $\boldsymbol{r_1}$, $\boldsymbol{r_2}$ and $\boldsymbol{r_4}$ in addition to $\boldsymbol{r_0}$ which acts as the identity. Hence $\boldsymbol{r_5}$ is not generated by $\boldsymbol{r_2}$ and $\boldsymbol{r_3}$ (in spite that 5 = 2 + 3). Also note that this numbering holds only with that particular ordering of the $\boldsymbol{r_k}$ functions, known as the Hadamard ordering (Hurst *et al.* 1985; Beauchamp 1987). For other orderings, the property that n+1 functions generate the rest of the 2^n functions will still hold but the numbering of the functions will be different, i.e. those will no longer be the ones with the 2^k index.

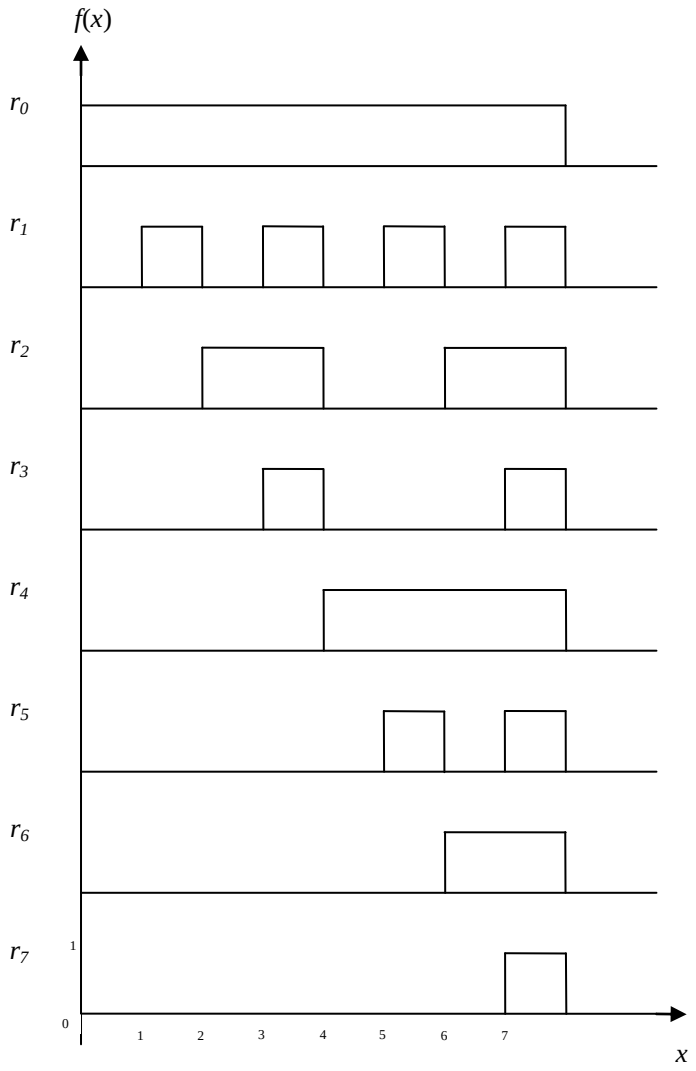


Figure 6-1: The Reed-Muller functions for three variables.

In summary, n+1 particular RM functions generate the rest of the 2^n ones by component-wise multiplication, and the total 2^n RM functions generate all 2^{2^n} binary functions by linear combination.

We have mentioned earlier that one of the benefits of representing the RM expansion as a linear transformation on a vector space is that it makes it easier to extend it to a larger number of variables. By comparing equation 6-14 and equation 6-6 we find that the transformation matrix for the three variables case (call it $\boldsymbol{T_3}$) relates to that for the two variable case (call it $\boldsymbol{T_2}$) by

$$\boldsymbol{T_3} = \begin{bmatrix} \boldsymbol{T_2} & \mathbf{0} \\ \boldsymbol{T_2} & \boldsymbol{T_2} \end{bmatrix} \qquad 6\text{-}17$$

where $\mathbf{0}$ in the matrix above is a 4 × 4 matrix of zeros.

For the general case of a function of n variables, the transformation matrix can be obtained in a recursive manner by (Green 1986; Almaini 1994)

$$\boldsymbol{T_n} = \begin{bmatrix} \boldsymbol{T_{n-1}} & \mathbf{0} \\ \boldsymbol{T_{n-1}} & \boldsymbol{T_{n-1}} \end{bmatrix} \qquad 6\text{-}18$$

where $\mathbf{0}$ is a matrix of zeros of dimensions ($2^{n-1} \times 2^{n-1}$) and $\boldsymbol{T_1}$ is given by

$$\boldsymbol{T_1} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad 6\text{-}19$$

In the previous chapter we have considered combinatorial gene regulatory functions of one and two regulatory variables. Using equations 6-18 and 6-19 the RM expansion (and hence transform) for a larger number of variables can be determined directly, which is much easier than the summation of equation 6-12. There are even simpler ways to represent this recurrence relation, one method uses what is known as the Kronecker matrix product whereby the product of two matrices is defined as the multiplication of the second matrix by every element of the first. This means that the Kronecker product of an $m \times n$ matrix by an $r \times s$ matrix is an $(mr) \times (ns)$ matrix. Using this notion, the transformation matrix $\boldsymbol{T_n}$ for an n variable function is given by the Kronecker product of the matrix $\boldsymbol{T_1}$ by itself n times (Green 1990a)

$$\boldsymbol{T_n} = \overset{n}{\otimes} \boldsymbol{T_1} \qquad 6\text{-}20$$

where the encircled multiplication sign indicates the Kronecker product of two matrices, and the *n* indicates how many times the multiplication will be performed. Using this notation, equations 6-6 and 6-14 above can be easily reproduced. This is an even simpler notation than equation 6-18 and produces faster computational algorithms (Green 1990a).

The RM expansion for the general *n* variable case corresponding to equation 6-18 is given by the linear combination of the RM functions r_k

$$f(x_n, ..., x_2, x_1) = \sum_{k=0}^{2^n - 1} a_k r_k \qquad 6\text{-}21$$

This equation can be used in the synthesis of logic functions, which constitutes the second benefit of the RM transform as alluded to earlier. We will explore this further in the context of synthetic biology.

6.3 Application to synthetic biology

Synthetic biology is a field of research that combines biological knowledge with engineering methodologies to produce biologically based systems that implement new biological functionalities or improve on existing one, (Endy 2005; Andrianantoandro *et al.* 2006). As an emerging interdisciplinary field it is still relatively in flux and has not completely morphed in terms of scope and tools. In such situations, it is not uncommon to have different perspectives of the field depending on the background of the researchers (Brent 2004; Benner and Sismour 2005; de Lorenzo and Danchin 2008; O'Malley *et al.* 2008). For scientists it is a tool for testing biological hypotheses, generating new ones and even attempting to create artificial life (Gibson *et al.* 2010). From an engineering perspective however, synthetic biology is another engineering discipline, but that uses biological "technology" for implementing the designed systems. Thus it employs biological components rather than physical ones such as electronic, mechanical or structural components.

Applying the engineering design methodology to the design of systems using biological components has produced several successful biological "devices" some with novel functionalities that do not exist in nature, and that may not necessarily be biologically useful but nonetheless serve to prove the methodology. Many of the devices (we use the term here in a generic sense to mean functional units) developed are catalogued in a standard way in an online registry [partregistry.com] that resembles the format of the data sheets of electronic components. Some of the notable examples of engineered biological systems have been reviewed by several authors (Heinemann and Panke 2006; Drubin *et al.* 2007; Marchisio and Stelling 2008). Indeed the design approach in synthetic biology has been particularly influenced by that of electronic engineering as evident in adopting such terminology as genetic circuits and logic design (Hasty *et al.* 2002), and where the design of different functionalities has mimicked logic design in electronic engineering.

As has been mentioned in chapters three and five, logic design is usually performed using the disjunctive normal form (DNF) of a logic function, which leads to implementation using the well known logic gates AND, OR and NOT. This is also the approach adopted in synthetic biology; indeed by abstracting from the implementation technology (electronic vs biological) to the function being implemented, one can transfer the methodology across disciplines. In the biological domain several authors have suggested implementations of those gates, usually employing transcription factors as inputs and mRNA as the output (Guet *et al.* 2002; Dueber *et al.* 2004; Rodrigo and Jaramillo 2007; van Hijum *et al.* 2009). In such a case the logic gate is effected using the cis-regulatory logic of the gene, see chapter three and Istrail and Davidson (2005). One of the problems with the DNF approach however, is the so called high fan-in required of the logic gates, meaning that a large number of inputs is required to be connected to each gate, especially in the case of a large number of variables. In the biological context this causes a problem of molecular overcrowding at the promoter of the gene (Buchler *et al.* 2003). Hence an alternative candidate for the design task in synthetic biology is the Reed-Muller expansion, in particular in its transform form. The RM transform in effect automates the design task and makes it transparent to the designing biologist who does not need to be concerned with the mathematical background involved. In addition the Reed-Muller formulation of a logic function allows for more implementation architectures

than are readily available for the DNF (Green and Edkins 1978). Another feature of the RM approach is that it allows for modelling incompletely specified functions, i.e. functions for which the outputs are only known for a subset of the possible input combinations. Such a situation is conceivable in the context of gene expression regulation where some regulatory factors might not have been tested for all their possible values (Habib 1993; McKenzie *et al.* 1993; Debnath and Sasao 2000). It should be noted that the DNF approach also allows for modelling such a situation, but not in as straightforward a way as it is in the RM approach.

However, the RM design approach is not without drawbacks, the most prominent is that it is difficult to obtain a minimal design, i.e. one that fulfils the specifications while maintaining a minimum number of components. This, together with the lack of development tools and the heavy capital and expertise already invested in the traditional design approach are some of the reasons why the RM approach has not been widely deployed in the electronics industry. The situation with synthetic biology however is different; the small scale of the designs makes it possible to go through an exhaustive search for the optimal design, or to utilise some of the somewhat difficult optimisation techniques of the RM expansion (Green 1990a; Green and Khuwaja 1992; Debnath and Sasao 2000; Falkowski and Yan 2004). Furthermore, being a nascent industry, there has not been heavy investment yet in productivity tools and expertise that would otherwise prohibit exploring alternative design approaches. Indeed, it is well recognised in industry that success of a method or a product is not always based on technical merit, but often on commercial and economic factors.

As is well known, in any design task, the designer is given a set of specifications and is required to produce a physical system that meets them. The specifications are given at different levels of abstraction, for example for an electronic circuit they start with the function to be performed by the system, but also include limits on power consumption, speed of response, size, weight, fault tolerance and so forth. Here we will limit our discussion to the functional specifications. Assume we are given the specification in the form of the output values desired for the different inputs, this is equivalent to being given the truth vector $\boldsymbol{d}$. Hence all the designer has to do is multiply the truth vector by the transformation matrix of the matching dimension as

in equation 6-7 to obtain the linear combination as in equation 6-21 which gives the weighting on the basis functions needed to construct the desired function.

Let us demonstrate the application of the RM transform method to synthetic biology by a fictitious example. Assume that a biologist wants to design a gene regulatory system that is controlled by two conditions, whose effects may possibly be mediated through transcription factors. It is required that each transcription factor on its own represses the gene, thus when neither is present the gene will be expressed. Assume further that it is also required that when both conditions are present at the same time, the gene is switched on, possibly due to both repressors cancelling each other's effect. Thus we can use the function specifications outlined above to build a truth table for this system as in table 6-2 where x_1 and x_2 represent the transcription factors and $f(x_2, x_1)$ the gene expression level.

Table 6-2: Specifications of a biological function to be synthesised.

x_2	x_1	$f(x_2, x_1)$
0	0	1
0	1	0
1	0	0
1	1	1

Using the truth values in the table in equation 6-6 above we get the coefficients of the RM expansion, substituting those in equation 6-5 we get

$$f(x_2, x_1) = 1 \oplus x_1 \oplus x_2 \qquad 6\text{-}22$$

Thus the biologist gets the required mathematical function right away, just by a simple matrix multiplication on GF(2). Such a procedure can easily be automated to become completely transparent to the user, whereby they enter the truth values and get the coefficients. In fact there are functions in the mathematical packages MATLAB and Mathematica that perform finite field arithmetic. The biological aspect of the problem however, is more challenging than the mathematical one, i.e. how can one implement this function using biological components. We will discuss this issue later in this chapter.

6.3.1 The stoichiometric matrix as a linear transformation

It is worth pointing out at this stage that this linear combination is conceptually similar to the stoichiometric matrix used in metabolic engineering for the analysis of the fluxes in a given metabolic pathway, or indeed the whole metabolic network. Metabolic engineering can be thought of as a precursor to synthetic biology in the sense that its purpose was also to alter the genetic make-up of the organism, but to achieve changes at the metabolic level to increase the production of certain desirable metabolites often by orders of magnitude (Bailey 1991; Nielsen 2001). Which part of a pathway to alter is decided based on the flux analysis of the different reactions in the pathway and the control analysis of the enzymes catalysing those reactions. The analysis often involves a material balance through the pathway to determine the rate of concentration change of the different metabolites as the fluxes through the different reactions of the pathway change. We review briefly the formulation of the stoichiometric matrix to demonstrate how it can be viewed as a linear transformation.

Assume we have m metabolites involved in n reactions, and that metabolite i is involved in reaction j with the stoichiometric coefficient α_{ij}, then a straightforward material balance of the metabolite i through all the reactions in the pathway gives

$$\frac{dx_i}{dt} = \alpha_{i1}v_1 + \alpha_{i1}v_1 + ... + \alpha_{ij}v_j + ... + \alpha_{in}v_n \qquad 6\text{-}23$$

Where x_i is the concentration of the metabolite i and v_j is the flux of reaction j. Of course a metabolite will not normally be involved in every reaction of the pathway, so some of the α_{ij}'s will be zero. The other stoichiometric coefficients will be either positive or negative depending on whether the metabolite is a substrate or a product of the reaction they relate to. Equation 6-23 indicates that the rate of change of metabolite i will be determined by its net flux through all the metabolic reactions in which it is involved, whether as a substrate or a product. This same procedure applies to all the metabolites in the pathway and hence can be represented in matrix form

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{S}\boldsymbol{v} \qquad 6\text{-}24$$

Where $\boldsymbol{x}$ is a vector of the concentrations of the m metabolites, hence it is m dimensional, $\boldsymbol{v}$ is the vector of fluxes through the n reactions, thus n dimensional and $\boldsymbol{S}$ is the $m \times n$ matrix of stoichiometric coefficients, given by

$$\boldsymbol{S} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & . & \alpha_{1j} & . & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & . & . & . & . \\ . & . & . & . & . & . \\ \alpha_{m1} & . & . & . & . & \alpha_{mn} \end{bmatrix} \qquad 6\text{-}25$$

Each row in the stoichiometric matrix $\boldsymbol{S}$ corresponds to a particular metabolite in the pathway, and every column corresponds to a reaction. So row *i* indicates all the reactions in which metabolite *i* is involved, while column *j* indicates all the metabolites involved in reaction *j*. Normally the number of reactions is larger than the number of metabolites because some metabolites are involved in several reactions, a notable example of that are metabolic precursors which are the starting points for many pathways. Hence the stoichiometric matrix has more columns than rows, i.e. $n > m$.

We have mentioned earlier the well known algebraic fact that any matrix can represent a linear transformation between two vector spaces, and the stoichiometric matrix is no exception. It transforms the space of reactions (fluxes) to the space of rates of changes of metabolite concentrations, with the transformation given by equation 6-24. In analogy with equations 6-14 and 6-16 above, equation 6-24 can be written in the form

$$\frac{d\boldsymbol{x}}{dt} = \begin{bmatrix} \boldsymbol{s}_1 & \boldsymbol{s}_2 & . & \boldsymbol{s}_j & . & \boldsymbol{s}_n \end{bmatrix} \boldsymbol{v} \qquad 6\text{-}26$$

Where the $\boldsymbol{s_j}$'s are the reaction vectors containing the stoichiometric coefficients of all the metabolites involved in the corresponding reactions. Again in analogy with equation 6-21, equation 6-26 can be written as

$$\frac{d\boldsymbol{x}}{dt} = \sum_{j=1}^{n} v_j \boldsymbol{s}_j \qquad 6\text{-}27$$

Since the space of reactions is *m* dimensional because each reaction can have a maximum of *m* metabolites, and since we have *n* reactions where $n > m$, it follows that not all the vectors $\boldsymbol{s_j}$ are linearly independent. The maximum possible rank of the stoichiometric matrix is *m*, however this is often not the case and the rank is less than *m*, because some of the reactions are usually linearly dependent, for example having

stoichiometric coefficients that are multiples of each others. This means that the basis set that spans the reaction space is a subset of the set of reaction vectors. The stoichiometric matrix is extensively studied by Palsson (2006) from an algebraic viewpoint with interesting metabolic implications.

Our purpose from this discussion of the stoichiometric matrix and the transform form of the Reed-Muller expansion is to demonstrate that the concept of a linear transformation between two vector spaces is applicable in the biological context with the relevant interpretation of the spaces. The two contexts we used here are that of gene expression regulation and of metabolic flux analysis.

It is clear from this discussion that a linear transformation can be non-square and hence not invertible such as the stoichiometric matrix, or it can be square and invertible but non-orthogonal such as the Reed-Muller transformation matrix. A further case is when the invertible transformation is also orthogonal. Orthogonal transformations have appealing features as their basis functions are intuitively similar to the axes of a Euclidean space, but more importantly they have computational advantages over non-orthogonal ones.

For a real function, the orthogonal transform most familiar to engineers is the Fourier transform, but there are others, depending on the characteristics of the function being transformed. There are also several orthogonal transforms for binary functions, the most common and arguably intuitive one is the Walsh transform because of its simplicity (Beauchamp 1975). The Walsh transform can be applied to binary functions but requires first the transformation of the binary set {0, 1} to the binary set {1, -1}. Algebraic equivalence between the two under certain binary operations can be established, but we will not pursue this further.

6.4 Extension to the multiple-valued case

Recall that one of the aims of this work is to develop a mathematical modelling method to represent discrete gene regulatory function that can take more than two values. Toward this end we first addressed the two valued case using the Reed-Muller expansion which is essentially an expansion on the two element finite field GF(2).

Whilst the simplest finite field, there is nothing fundamentally special about GF(2), and hence the method can be applied to any finite field. Consequently, many of the concepts introduced in the previous chapter and the earlier sections of this chapter can be extended to the multiple-valued case with similar interpretations.

6.4.1 Functions on finite fields

Let us start by going back to the basic argument behind this work, which might have been obscured in the discussion of the binary case. On an abstract level, the main idea is that a biomolecular system (or any other system for that matter) that has multiple discrete states, when appropriately defined can be described as a mapping from one finite set to another, where the number of elements of the set corresponds to the number of discrete states of the biomolecular entities involved. This mapping can often be represented as a function on some algebraic structure, e.g. a group, a ring or a field. Now to move to a more concrete argument, the biomolecular system in our case is the regulation of gene expression and the algebraic structure is a finite field.

A powerful property of a finite field is that any function on it can be uniquely formulated as a polynomial on the field with coefficients from the field. Furthermore, the degree of this polynomial is less than the order of the field, i.e. less than the number of discrete values of the variable involved. This has been discussed in chapter four where we have indicated that the order q of a finite field must be either a prime or a positive integer power of a prime. When the order is a prime p, modular arithmetic is used. However when it is a power of a prime p^n then modular arithmetic can no longer be used and addition and multiplication have to be defined differently (Berlekamp 1968; Lin and Costello 1983; McEliece 1987). To see this, consider the case of GF(4), where table 6-3 gives the modulo 4 operations while table 6-4 the GF(4) operations which are clearly different.

Table 6-3: Addition and multiplication modulo 4.

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

×	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Table 6-4: Addition and multiplication on GF(4).

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

×	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

It is clear from table 6-4 that elements of a finite field can no longer be treated as ordinary numbers, rather they are more general mathematical entities, and that is why they are often denoted by symbols rather than numbers (in fact they can be regarded as polynomials). What matters for a finite field is not the nature of its elements but the structure of the field. Indeed all finite fields of a given order are isomorphic (mathematically equivalent), irrespective of how the elements and the two binary operations are defined. This is a powerful property that allows us to redefine the elements to suit whatever context we are using the field to model. See chapter four for further discussion of finite fields.

Recall that we are modelling gene regulatory functions as combinatorial logic functions, now defined on a general Galois field GF(q) rather than on the two element GF(2). The theory of logic functions on finite fields is well developed in electronic engineering where it is termed multiple-valued logic, and has been around for some time (Menger 1969; Benjauthrit and Reed 1976, 1978; Pradhan 1978). The motivation for developing such tools for logic design was the optimisation of several cost factors in logic circuit design such as the number of gates, utilisation of microchip area, switching speed and testability (Falkowski and Lozano 2005). In spite of its potential benefits, multiple-valued logic design did not gain wide acceptance in the digital design community due to the lack of efficient design tools and implementation technology, among other reasons (McCluskey 1986). However, we believe that it can prove valuable in the modelling of multiple-valued discrete gene regulatory functions and can be used both as an analysis and a synthesis tool.

A q-valued function of a q-valued variable can be modelled as a polynomial on GF(q) by

$$f(x) = \sum_{k=0}^{q-1} a_k x^k$$
$$= a_0 \oplus a_1 x \oplus a_2 x^2 \oplus .... \oplus a_{q-1} x^{q-1} \qquad 6\text{-}28$$

This is sometimes referred to as the multiple-valued Reed-Muller expansion. Note that now the ring sum sign denotes addition on GF(q) and all the quantities in the equation, whether the coefficient a_k or the variable x and its different powers are q-valued. The proof of this result from an electronic engineering perspective can be found in the sources cited above (Menger 1969; Benjauthrit and Reed 1976, 1978; Pradhan 1978), and from a mathematical perspective in Lidl (1994).

The counterpart of equation 6-28 in the binary case was given three different interpretations namely a function on a Boolean algebra and a polynomial on a finite field both covered in the previous chapter, and a transform on a function space discussed above. Recall from chapter four that a Boolean algebra is a special case of a distributive complemented lattice, hence by posing the multiple-valued case in such a framework we can give equation 6-28 a corresponding interpretation, however this requires a generalisation of the notions of complement, logic AND and OR. Such generalisations do exist (Green 1986) but the mathematics becomes awkward, and more importantly there is no obvious benefit from this interpretation. The second interpretation, namely as a polynomial on the finite field GF(q) has already been covered above. Interestingly, in analogy to the binary case, the concept of a difference operator on a finite field, akin to the Boolean difference of the previous chapter has been proposed by several authors leading to MacLuaren and Taylor series types of expansions for functions of several variables on GF(q), (Thayse 1974; Wesselkamper 1978; Hwan Mook *et al.* 1998; Stankovic *et al.* 2004). However it is the transform view of the expansion that we are interested in in this chapter, and we will introduce it next.

As usual, let us take a concrete example, say q = 3 known as ternary logic. This means that we will use modulo 3 operations as indicated in table 6-5 which makes the derivation easier to follow. In such a case, equation 6-28 becomes

$$f(x) = a_0 \oplus a_1 x \oplus a_2 x^2 \qquad 6\text{-}29$$

Table 6-5: Addition and multiplication modulo 3.

+	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

×	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

We follow a derivation similar to the one used in chapter five, but now for a ternary function defined by the values of table 6-6, and presented by Green (1989).

Table 6-6: Truth table for a ternary function

Input *x*	**Output *f*(*x*)**
0	d_0
1	d_1
2	d_2

Substituting the different values of x and the corresponding values of $f(x)$ from table 6-6 into equation 6-29 and noting that we are using modulo 3 arithmetic, we get

$$\begin{aligned} d_0 &= f(0) = a_0 \\ d_1 &= f(1) = a_0 \oplus a_1 \oplus a_2 \\ d_2 &= f(2) = a_0 \oplus 2a_1 \oplus a_2 \end{aligned} \qquad 6\text{-}30$$

or in matrix form

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \qquad 6\text{-}31$$

By manipulating equations 6-30 or by inverting the matrix in equation 6-31 and again remembering that we are using modulo 3 arithmetic, we get

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} \qquad 6\text{-}32$$

or in compact form

$$\boldsymbol{a} = \boldsymbol{T}\boldsymbol{d} \qquad 6\text{-}33$$

and correspondingly

$$\boldsymbol{d} = \boldsymbol{T}^{-1}\boldsymbol{a} \qquad 6\text{-}34$$

It is clear from equations 6-31 and 6-32 that now the transform matrix $\boldsymbol{T}$ and its inverse are different, unlike the binary case (see chapter five).

Again following the presentation of the two valued case in chapter five, for a function of n variables that are q-valued we get a polynomial on GF(q) with q^n terms,

$$f(x_n,...,x_2,x_1) = \sum_{k=0}^{q^n-1} a_k x_n^{q_{kn}} ... x_2^{q_{k2}} x_1^{q_{k1}} \qquad 6\text{-}35$$

This is similar to equation 6-12 above, but now all the coefficients and variables are q-valued and the summation is over GF(q). Also now $q_{kn}....q_{k2}q_{k1}$ is the n digit q-uary (i.e. base q) representation of the decimal digit k whose values range from 0 to $q^n - 1$. The highest possible power of any particular variable in this polynomial is less than q, i.e. it is less than or equal to $(q - 1)$, and consequently the highest possible power of any monomial (i.e. product of variables) is $n.(q - 1)$. Again we will be faced with the issue of the ordering of the variables, and we will follow the same convention as with the binary case above, but now the order will be that of counting in base q. Let us take the example of a two variable ternary function, whereby we get

$$f(x_2,x_1) = a_0 \oplus a_1x_1 \oplus a_2x_1^{\,2} \oplus a_3x_2 \oplus a_4x_2x_1 \oplus a_5x_2x_1^{\,2} \oplus a_6x_2^{\,2} \oplus a_7x_2^{\,2}x_1 \oplus a_8x_2^{\,2}x_1^{\,2} \qquad 6\text{-}36$$

We have 9 (= 3^2) terms, the highest power of any variable is 2 (= 3 – 1), and the ordering of the powers of the variables correspond to counting in base three as explained in table 6-7.

Table 6-7: Ordering of the variables for the two variable ternary function of equation 6-36. MSD = Most Significant Digit, LSD = Least Significant Digit.

Term number in equation 6-36	Corresponding ternary no.		Term in equation 6-36	
	MSD	LSD	Powers of x	Monomial
0	0	0	$x_2^0 x_1^0$	1
1	0	1	$x_2^0 x_1^1$	x_1
2	0	2	$x_2^0 x_1^2$	x_1^2
3	1	0	$x_2^1 x_1^0$	x_2
4	1	1	$x_2^1 x_1^1$	$x_2 x_1$
5	1	2	$x_2^1 x_1^2$	$x_2 x_1^2$
6	2	0	$x_2^2 x_1^0$	x_2^2
7	2	1	$x_2^2 x_1^1$	$x_2^2 x_1$
8	2	2	$x_2^2 x_1^2$	$x_2^2 x_1^2$

The two variable ternary transform can be obtained from the one variable one in a recursive manner, similar to the binary case (Green 1989), where $\boldsymbol{T_1}$ is the 3 × 3 matrix in equation 6-32 above. In general for the *n* variable case we have

$$\boldsymbol{T_n} = \begin{bmatrix} \boldsymbol{T_{n-1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 2\boldsymbol{T_{n-1}} & \boldsymbol{T_{n-1}} \\ 2\boldsymbol{T_{n-1}} & 2\boldsymbol{T_{n-1}} & 2\boldsymbol{T_{n-1}} \end{bmatrix} \qquad 6\text{-}37$$

A readable and fairly comprehensive treatment of the general multiple-valued case from an engineering perspective, including the direct and inverse transformation matrices, some recursive relations and the arithmetic operations on several finite fields is given by Green and Taylor (1974). Computational aspects of the problem of determining the coefficients in the expansion have been addressed by several authors, for example (Green 1989, 1990b; Jankovic *et al.* 2002; Falkowski and Lozano 2005; Falkowski *et al.* 2005). Similarly, optimisation of the multiple-valued case including incompletely specified functions is addressed by (Green and Edkins 1978; Watanabe and Brayton 1993; Yunjian and Brayton 2000).

Note that the matrices discussed above are linear transformations on vector spaces where the columns of the matrix are linearly independent vectors, hence leading to invertible matrices. We will not derive these results as they are essentially mechanistic extensions of the binary case, instead we will discuss later some of the conceptual issues underlying transforms and how they can be interpreted in the

biological context. First we consider the application of multiple-valued logic to synthetic biology design.

6.5 Synthetic biology using multiple-valued logic

As mentioned earlier design in synthetic biology mimics logic design in electronic engineering. Logic design is mostly based on binary logic due to its intuitive relationship to the binary states of electronic switches, ON and OFF or abstractly 1 and 0. The same mind set has been carried over to logic design using biomolecular components in spite of them being free from this restriction. However, we believe that this need not be the case, and that other forms of logic can be utilised and are in fact more suitable for the description of logic systems based on biomolecules. Unlike electronic switches, biomolecules and biomolecular components in general can have more than two states. Hence, the type of logic employed in formulating the biological function will depend on the number of states of the "technology" or biomolecular components used to implement it. Working at a mathematical level of abstraction, the challenge then would be to formulate the biological function in the framework of multiple-valued logic.

In fact because of their inherent multiple state capabilities, biomolecular components can even be used to implement non-biological functions such as those in an arithmetic logic unit of a microprocessor. This leads to using fewer components and hence less delay and routing issues between the components. Furthermore, the same number of inputs can produce significantly larger number of functions. For example as outlined in table 6-1 above, two binary inputs can produce sixteen (= 2^{2^2}) different binary functions; on the other hand two ternary inputs can produce 19683 (= 3^{3^2}) different ternary functions. This means that increasing the logic levels by just one, from binary to ternary, results in a huge increase in the number of possible functions.

Thus when designing biological systems that implement non-biological functions (as opposed to modelling existing biological ones), such as an adder for example, there are several challenges faced. First is the choice of the logic levels; this will be dictated by the context of the problem and will lead to the mathematical formulation

of the required function as mentioned above. Next comes the implementation issues such as the choice of biomolecular components with the required number of states and the biomolecular mechanisms that will execute the function.

For example if one wants to design a quaternary (radix 4) adder, then the natural choice for logic levels is four. Next we need to formulate the function implementing the adder, this is straight forward from the truth table of the adder and the Galois field approach outlined above, where in this case GF(4) would be used. Then we need to choose a biomolecule with four states, a suitable candidate would be a nucleotide, which abstractly can be viewed as a variable that can take one of four different values namely A, T, C and G. Next we need to identify the different "values" of the nucleotide (as a quaternary) variable with elements of the field GF(4); recall the discussion above about the abstract nature of the elements of a field. We had developed one such mapping, albeit in a different context but is applicable here, see Appendix I (Aleem *et al.* 2009). Perhaps the greatest challenge in this particular problem is how to implement the mechanism that will read the strings of nucleotides representing the quaternary numbers to be added, perform a bit–wise addition on them and give the output. This can involve RNA polymerase to read the strings and some other mechanism for addition, and outputting. Such an implementation is at a different level of abstraction, mainly closer to the biological level than to the mathematical one we are interested in in this work. It can be considered to fall in the realm of DNA computing, which is essentially computation using DNA molecules. This is a multi-disciplinary field with research relating to both the mathematical/computational aspect of DNA computing and the biological aspects. The former addresses such issues as the types of problems that can be solved and the performance of the algorithms implemented, while the latter considers issues such as the design of the DNA sequences involved and the different molecular manipulations required for implementing the computations (Amos 2005). In fact the term biomolecular computing is starting now to replace the term DNA computing to indicate that other biomolecules such as RNA are used. Other examples of biomolecules that can have several states include a morphogen which can have several concentration thresholds each triggering a different behaviour in a developing cell (see chapter three), and a regulatory protein which can have several activation states depending on how many activation sites it has.

Whilst we do not rule out a binary logic approach to design since many biomolecules do act in a binary fashion, we maintain that it is by no means the only or possibly even the best option in some cases. Such natural deviation in behaviour from the simple ON/OFF switches should be exploited in the design process.

6.6 A conceptual view of transforms

In this section we want to offer a brief discussion of the idea behind a transform, relying on an intuitive argument rather than mathematical derivations. The first question that comes to mind concerning transforms is, why do we need them in the first place? In other words, what is the benefit we gain from using a transform in analysing a problem? Well, the main benefit is that it transforms the information about the function from one form into another (technically termed from one domain into another), with the hope that the information in the new form will reveal some features or characteristics of the function that are difficult to discern in the original form. An example familiar to engineers is the Fourier transform which transforms the information in a signal from the time domain to the frequency domain, a process known as harmonic analysis. This means that the time course of the function reveals information about its frequency content. This is very helpful in studying many problems in engineering where frequency determines the response of the system. For example electronic engineers use harmonic analysis for the analysis and design of communication systems and their components such as filters and tuners. Mechanical engineers use it for vibration analysis of rotary machinery, and civil engineers use it for the analysis of structures under dynamic loads, to name but a few examples.

For a transform to be useful it must retain all the information that is in the original function when it transforms it into the new form (or domain), otherwise some of the features that we are hoping to detect in the new domain might get lost in the process, rendering the exercise useless. Intuitively this means that since all the information is intact in the new domain, it should be possible to recover it back to the original domain; in a sense reversing the process. Mathematically this means that the transformation, which is a matrix in our context here, must be invertible. Now, we know from algebra that for a matrix to be invertible it has to have full rank, and this only happens when its columns (and rows) are linearly independent, hence our

emphasis earlier on linear independence. Note that orthogonality is not necessary for linear independence (although it is sufficient). Our interest in orthogonality is mainly for convenience since as mentioned earlier, it provides computational advantages and gives an intuitive feel to the transform.

There is another interesting facet of invertible transforms that comes from information theory. Information theory started as a branch of communication engineering that was concerned with the information content in a signal and whether it is possible to recover all the original information in the transmitted signal from the information in the received signal, given that it had travelled through a noisy communication channel and hence corrupted (Shannon 1948; Cover and Thomas 1991). The noisy channel was characterised by an error probability distribution, meaning that different parts of the signal (e.g. frequencies) will have different probabilities of error. This characterisation was used to calculate the information content of the received signal (which has travelled through the noisy channel) using a function that was called the entropy of the signal. This term was used because of the similarity in the form of the function to that of entropy in statistical thermodynamics. Indeed both entropies in a sense carry information about the reversibility of the process. This means that a transform such as the Reed-Muller or Fourier or any reversible transform in general, represents a constant entropy process from the point of view of information content. This matches perfectly with our earlier argument about the reversibility of the transform as a condition for not losing any information, i.e. for being able to recover all the original information, hence reversing the process. The transform and its inverse are often referred to as the transform pair, and they have even more interesting implications. Which one is the direct transform and which is the inverse does not matter conceptually since both contain the same information albeit in different forms. Nonetheless there are certain conventions, for example in the Fourier transform pair, the transform from time to frequency is considered the direct transform and from frequency to time is the inverse transform. For the Reed-Muller transform we will consider the transform from the truth table domain to the polynomial domain as the direct transform, and the reverse direction the inverse one, such as in equations 6-33 and 6-34 above. The direct transform of equation 6-33 uses the truth values of the function given by the vector $\boldsymbol{d}$ to determine the vector $\boldsymbol{a}$ of the coefficients in equation 6-29. This means that it determines the contribution of the

different terms (representing the basis functions) to the overall function, i.e. it analyses the function to its constituents. This is the same approach of the Fourier series where it uses the time function to determine the contributions of the different harmonics. Recall that the Fourier series (which is a special case of the Fourier transform) is used to breakdown a periodic function into its harmonics and is given by (Kraniauskas 1992)

$$f(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos k\omega_0 t + b_k \sin k\omega_0) \qquad 6\text{-}38$$

Where ω_0 is the fundamental frequency of the periodic signal and τ_0 is its period. The different coefficients (the amplitudes of the waves) are given by

$$\begin{aligned} a_0 &= \frac{1}{\tau_0} \int_0^{T_0} f(t) dt \\ a_k &= \frac{2}{\tau_0} \int_0^{T_0} f(t) \cos k\omega_0 t dt \\ b_k &= \frac{2}{\tau_0} \int_0^{T_0} f(t) \sin k\omega_0 t dt \end{aligned} \qquad 6\text{-}39$$

We have used the Fourier series rather than the more general Fourier transform merely for simplicity of presentation, and because it is in the form of a sum like the case of the RM expansion. The frequencies are integer multiples of the main frequency of the periodic function, i.e. they are its harmonics, hence the term harmonic analysis. The function is said to be expanded in terms of the cosines and sines (which are the basis functions for this case) where equations 6-39 determine the coefficients of the expansion representing the contributions of the different harmonics to the overall function as in 6-38. Equations 6-39 represent the direct transform, and is known as the analysis transform as it analyses or resolves $f(t)$ to its constituents. On the other hand equation 6-38 represents the inverse transform and is known as the synthesis transform as it uses the different basis functions to build the time function.

A similar interpretation applies to the Reed-Muller expansion where equation 6-33 represents the direct or analysis transform as it breaks down the function given by d to the contributions from the different powers of x as in equation 6-29. Meanwhile the synthesis equation 6-34 or equivalently 6-29, uses the basis functions as specified by

the vector $\boldsymbol{a}$ to build the function vector $\boldsymbol{d}$. So essentially the analysis transform says: here is a function, find out what it is made of? In systems biology this is known as a top-down approach (Westerhoff *et al.* 2009). Conversely the inverse transforms says: here is a set of functions, if you combine them with the coefficients (weights) given, what function do you end up with? Again in systems biology this is known as a bottom-up approach (Westerhoff *et al.* 2009). Hence the transform approach ties very elegantly with concepts from systems biology.

One final note we want to make from equation 6-38 and 6-39 above, and that is the direct transform of equations 6-39 uses all of the information in the time function to determine the contribution of every single harmonic. This is evident from the limits of integration which cover the whole period of the function, i.e. it includes all the information in the time domain. On the other hand the inverse transform uses all the information in the frequency domain to build the function in the time domain as evident from the limit of the summation which goes to infinity covering all the harmonics.

The situation with the RM expansion is a bit different, in that the direct transform uses some but not all the information for a given coefficient, only the last coefficient uses all the information as in equation 6-32. Similarly for the inverse transform in equation 6-31. It should be noted that there are other discrete transforms with this property, for example for the Walsh transform alluded to earlier, every coefficient in one domain carries information about the function at all its points in the other domain, as in the Fourier transform.

6.7 Summary and Conclusion

The core idea of this chapter is to apply the concept of transforms on function spaces to the case of discrete combinatorial gene regulatory functions. We started with the binary case building on the material of the previous chapter, where now we viewed the Reed-Muller expansion as a transform on a binary function space. The basis functions of this space are the Reed-Muller functions. For n binary variables, there are 2^n RM functions that span the space of all possible n variable binary functions; hence any such function can be constructed using the RM functions.

We then extended the concept of a transform on a function space to the multiple-valued case. This required first defining the functions that constitute this space, and for that we used a powerful property of finite fields whereby every function on the field can be uniquely represented by a polynomial on the field. Similar to the binary case this resulted in a number of functions that span the space and hence can be used to build any function on it. For the finite field GF(q), the number of functions is q^n where n is again the number of variables. Of course for both cases, the binary and multiple-valued, a set of functions that span a space has to be linearly independent.
We have also presented an interesting conceptual discussion of transforms in general that tied material from different areas of knowledge including information theory, statistical thermodynamics, communication engineering and systems biology, thus demonstrating the immense power of abstractions and of carrying concepts and tools across disciplines.

Interesting as it may be in its own right, a conceptual value is not sufficient to adopt a new method; practical benefits have to be accrued as well. The benefit can be in both modelling and design of combinatorial gene regulatory functions or biomolecular systems in general. The two concepts of modelling and design in the modern biological context correspond roughly to systems and synthetic biology respectively. We have addressed modelling implicitly in the previous chapter and design more explicitly in this one.

Indeed for the case of synthetic biology, using a multiple-valued design approach would decrease the number of components required to implement a function and consequently the material transport involved and the associated delay. In addition it can implement functionalities that may be difficult to achieve using binary logic. However, as with any new approach, just as there are opportunities there are challenges as well. For a method based on finite fields there are three main challenges. The first is the order of the field which has to be either a prime or a power of a prime, this is not a major issue since among the first nine integers (1 to 9) only 6 is neither; and it is unlikely to design a function with more than ten discrete levels (0 to 9). The second challenge is that of identifying members of a finite field with some biomolecule or biomolecular component. Again this is not a major issue as demonstrated above, as one is bound to find some biomolecule with the necessary

number of discrete states. Manipulating such a molecule then becomes the issue, and indeed this is the third and major challenge confronting such a design approach. There have been suggestions for implementing multiple-valued logic functions in electronics, which may be possible to emulate in biomolecular technology. However, this remains to be an area of potential future research.

This chapter concludes the development and interpretation of the method, we now want to apply it to a real life system in more depth than has been done in this and the previous chapter. The reason we did not get into too much detail of the biology in the previous chapters is that we did not want the biology to obscure the maths. But now that the mathematical formulation is completed we can apply it to a more detailed biological system, which we will do in the next chapter.

Chapter 7: Application to the Modelling of Phage Lambda

7.1 Introduction

In this chapter we apply the concepts introduced in chapters five and six, to model gene regulation in the bacterial virus known as phage lambda. As in any modelling endeavour it is important to understand the system being modelled in order to be able to both formulate a meaningful model and better interpret the results. Thus we will start this chapter by presenting phage lambda and its gene regulation. This presentation will be rather biological in nature and builds upon the background material introduced in chapter two. Here we will mimic the approach taken in that chapter where the material was introduced through a series of questions. We first ask, what is phage lambda and what does it do? Answering this question, in particular the second part will lead us to expand on the discussion of the regulation of gene expression by providing more detail on the molecular interactions involved in this process. This will lay the foundation for answering the second question, namely how does phage lambda effect its response to the regulating factors? Armed with this knowledge we can then attack the modelling problem, where we will develop two models, one binary and the other multiple-valued. The difference between the two is not merely mathematical, but is also conceptual in nature, an issue that we will discuss.

It is important to remember that the purpose of this chapter is not to present phage lambda for its own sake, but rather in order to use it as an example for the modelling approach discussed in this report. Hence it will not be discussed in any more detail than is necessary for this task. Indeed phage lambda was chosen here because it has been used in the scientific community as a model system, along with the *lac* operon, for the study of the regulation of gene expression. We have discussed the *lac* operon in some detail in previous chapters and here we introduce phage lambda. It should be pointed out that the first part of this chapter is based to a considerable extent on the

excellent book by Ptashne and Gann (2004). We will conclude this chapter and indeed this work, with a discussion of the merits and drawbacks of our method, in light of the application above.

7.2 What is phage lambda and what does it do?

Phage lambda is a virus that attacks bacteria. Like most viruses it consists of a single DNA molecule (a chromosome) encased in a protein "coat". The phage infects the bacterium by injecting its chromosome in the host leaving the protein coat behind. The infection causes the bacterium to go into one of two possible regimes known as lysis and lysogeny, explained below (figure 7-1). Which regime it will go into will depend on the conditions in the surrounding environment. Lambda is known as a bacteriophage i.e. bacteria eating, because it eventually destroys the bacterium it infects.

Lytic route

In the lytic response to infection, the lambda chromosome is replicated and the protein coat it encodes is synthesised extensively using the host bacterium transcription machinery. The bacterium becomes quickly filled with phage lambda viruses (DNA enclosed in protein) and after about 45 minutes the bacterium lyses (breaks down) and nearly 100 new lambda phages are released.

Lysogenic route

In the lysogenic regime, the lambda chromosome is integrated into the host bacterium DNA and is replicated and distributed passively with it as the bacterium grows and divides. In this case the lambda phage is known as a prophage. This is a stable situation that can go on for a long time if undisturbed. However, if the bacterium is irradiated with ultra violet radiation, it stops growing and about 90 minutes later it lyses and lambda phages are released.

The decision by lambda whether to lyse or lysogenise the bacterium depends on conditions in the surrounding environment. Normally if the nutrients are scarce, the bacterium will be deficient in the components required for the rapid and extensive lytic growth in which lambda is synthesised, hence lambda lysogenises the bacterium.

Phage lambda is often referred to as a genetic switch as it can switch its effect on the host bacterium from lysogeny to lysis in a process known as induction.

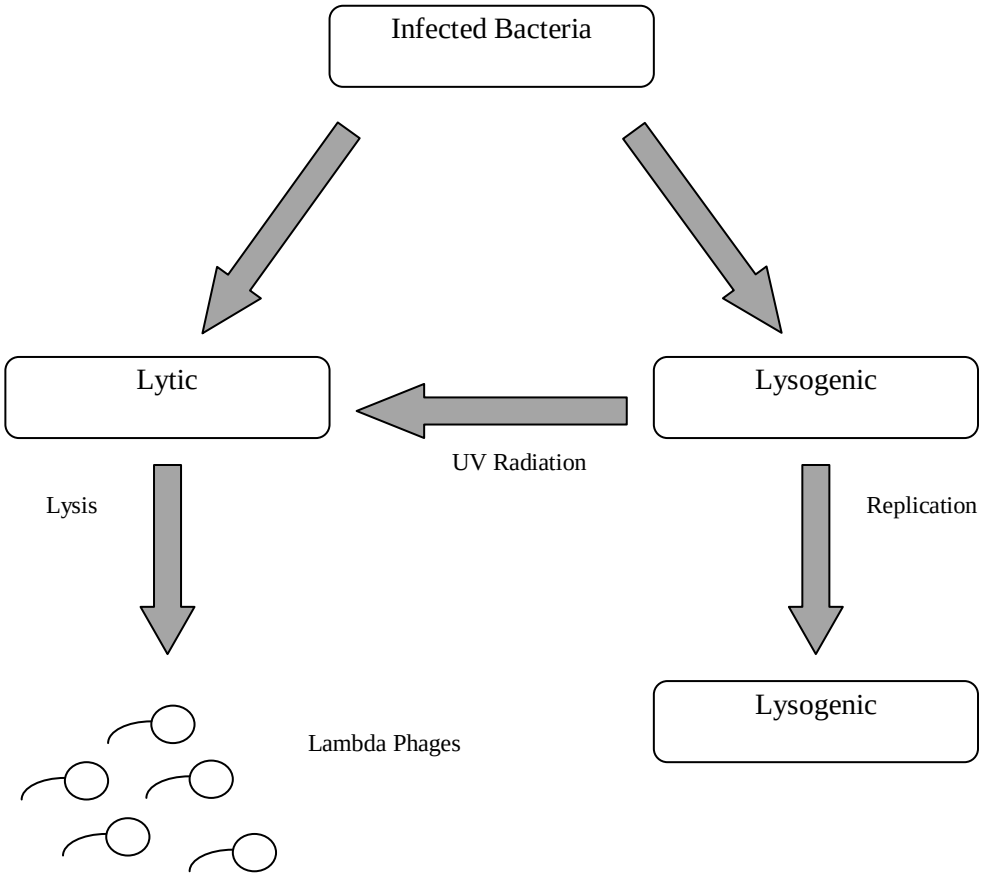


Figure 7-1: The two possible fates of a bacterium infected by phage lambda.

So how does lambda implement its effects? The answer is through the regulation of the right genes in its DNA (not the bacterium's). Before we explain the details of this however, we briefly review the control of gene expression at the molecular level as this will help us elucidate how lambda implements its function and consequently help us in our modelling task. This review will build on material already covered in chapter two, some of which will be repeated here for convenience and to make it self contained.

7.2.1 Molecular interactions regulating gene expression

Recall from chapter two that a gene is a long stretch of DNA that encodes for a protein. The expression of a gene starts with its transcription which makes a copy of the code in the form of RNA. This process is performed by the enzyme RNA polymerase and is often controlled by regulatory proteins. The first step in transcription is the identification of the starting site. This is done with the help of the promoter, which is a region of the DNA with certain sequence patterns that indicate to RNA polymerase that it is a promoter of a gene. It specifies the start site for transcription and the direction in which to proceed.

There are two important components involved in the regulation of gene expression, DNA sites and proteins. The interaction of proteins with each other and with certain DNA sites determines whether the gene will be transcribed or not and the rate of this transcription. We can envisage several scenarios for these interactions.

The most straightforward scenario is the interaction of the protein RNA polymerase with the promoter DNA site just described. However, the role of RNA polymerase in transcription can be facilitated or impeded by regulatory proteins that bind to other sites on the DNA called operators. For example a regulatory protein can bind to the operator region (which sometimes overlaps with the promoter) to block the access of RNA polymerase to the promoter hence preventing transcription, a mechanism known as exclusion. An example of this scenario is the lac repressor protein described in chapter two. On the other hand the regulatory protein can also bind to the operator region to help RNA polymerase bind to the promoter, a mechanism known as recruitment. An example of this scenario is the Catabolite Activation Protein (CAP) of the *lac* operon, again mentioned in chapter two although its specific action was not detailed there. Another scenario is when a regulatory protein binds to the operator site to recruit another regulatory protein that then helps recruit RNA polymerase, a mechanism known as cooperativity. Those four scenarios are summarised in table 7-1.

Table 7-1: Some scenarios of DNA/protein and protein/protein interactions.

Scenario	Protein	DNA site	Effect	Mechanism
1	RNA polymerase	Promoter	Straightforward Transcription	-
2	Regulatory protein	Operator or promoter	Prevent transcription	Exclusion
3	Regulatory protein & RNA polymerase	Operator (for reg. Prot), promoter (for RNA pol.)	Help start transcription	Recruitment
4	Two regulatory proteins	Two operator sites	Help recruit RNA polymerase to start transcription	Cooperativity

This discussion raises two questions

1. What determines whether RNA polymerase needs the help of a regulatory protein for recruitment to the promoter, or not?
2. What determines the binding of a given protein to the operator (whether for exclusion, recruitment or cooperativity)?

To answer the first question, we note that the structure of a promoter includes two standard sequences, known as consensus sequences, located at positions -10 and -35 upstream of the transcription starting site of a gene. RNA polymerase identifies those two sequences and binds to the promoter accordingly. In some cases the sequences deviate from the consensus pattern and hence it becomes difficult for RNA polymerase to bind to them. In such a case RNA polymerase will need the help of a regulatory protein to aid it in binding to the promoter, i.e. a case of recruitment. The closer the sequence of the promoter is to the consensus sequence the less it will need a regulatory protein to help bind RNA polymerase.

For the second question, the answer is determined by the affinity of the operator site to the protein, which is determined by both the sequence of the site and the shape of the protein. Since different operator sites normally have different sequences, they will have different affinities to a given protein.

At low concentration, the protein will bind to the site that has the highest affinity to it. As the protein concentration increases, it will bind to the site with the next lower

affinity, and so on. This type of affinity is known as the intrinsic affinity of the site to the protein, in the sense that it is determined by the site and protein structures. However, in some cases this intrinsic affinity can be altered with the help of another molecule of the same or different protein. For example sometimes when a molecule of a protein binds to one operator site it facilitates the binding of another of its molecules to another site of a lower affinity, even though the protein concentration may be lower than what would normally be needed for binding to that second site. This means that the protein/protein interaction in this case increases the effective affinity of the site to the new protein, i.e. a case of cooperativity.

In summary we can state the following:

a. A protein can bind to more than one operator site (usually not at the same time).
b. An operator site can bind more than one protein (again not at the same time).
c. Which site the protein will bind to (in either of a or b above) is determined by:
    - affinity of the site to the protein.
    - concentration of the protein.
    - cooperativity with another protein on an adjacent site.
d. When a protein binds to an operator site that overlaps with a promoter, it prevents RNA polymerase from binding to the promoter, hence switching the gene off or preventing it from turning on. This is known as negative control, and this particular mechanism is known as the principle of exclusion.
e. When the binding of the protein to the operator site helps RNA polymerase to bind to the promoter thus activating the gene, this is known as positive control, and this particular mechanism is known as recruitment.
f. When the binding of a protein molecule to the operator site helps another protein molecule to bind to another operator site, this is known as cooperativity.

Armed with this review of the molecular interactions involved in regulating transcription of a gene, we can now address gene regulation in phage lambda.

7.3 How does phage lambda control its course of action?

The switching of phage lambda from the lysogenic to the lytic route is controlled by two regulatory proteins that are encoded by two lambda genes. One protein is simply known as "repressor" and is encoded by a gene called *cI*, and the second is known as cro and is encoded by a gene called *cro*. In order to understand how the two proteins control lambda operation we need to consider the construction of the promoters and operators of the genes that the two proteins regulate, which are the same genes *cI* and *cro*.

7.3.1 Construction of the switching region

The region of the two promoters for the two genes *cI* and *cro* on the lambda DNA molecule is depicted in figure 7-2. It consists of the following sites:

1. P_{RM} the promoter for gene *cI* (shaded area on the left in figure 7-2)
2. P_R the promoter for gene *cro* (shaded area on the right in figure 7-2)
3. An operator region that is divided into three sites
   a. O_R1 overlapping with the promoter P_R
   b. O_R2 overlapping with both promoters P_R and P_{RM}
   c. O_R3 overlapping with the promoter P_{RM}

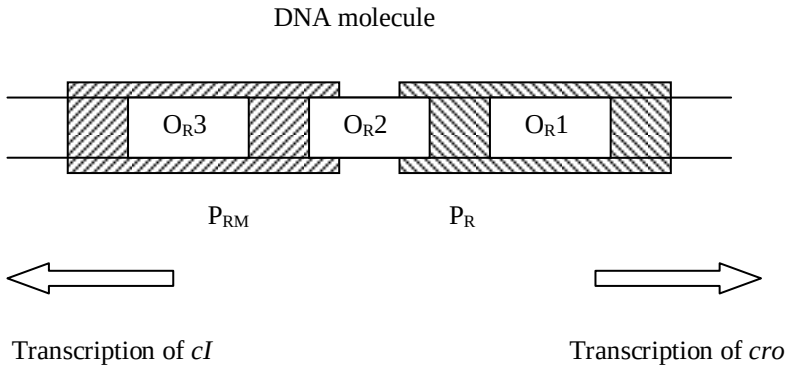


Figure 7-2: Part of the Lambda DNA molecule depicting the promoters and operator for the Lambda genes *cI* and *cro*.

The two genes encoding the regulatory proteins are not shown in the figure; only the promoters and relevant operator sites are. Note that the two promoters are adjacent, which means that the two genes transcribe in different directions as depicted in figure

7-2. It may be instructive to compare this figure with figures 2-5 and 2-6 in chapter two depicting the detailed construction of the *lac* operon.

A note on nomenclature

P_R = **R**ight **P**romoter, it is the promoter for the *cro* gene

O_R = **R**ight **O**perator

cro = **c**ontrol of **r**epressor and **o**ther genes, a regulatory protein expressed by the gene *cro*

P_{RM} = **P**romoter for **R**epressor **M**aintenance, is the promoter for the *cI* gene which expresses the regulatory protein "repressor"

repressor = a regulatory protein expressed by the gene *cI*

As a matter of convention, genes names are italicised while names of the corresponding (or other) proteins are not. Also note that the promoter P_{RM} "maintains" the level of "repressor" but it does not initiate its expression. This is done by another promoter for the same gene, but is not shown in this figure.

In lysogeny, all the lambda genes except *cI* are turned off to allow the lambda chromosome to replicate passively with the host DNA. The regulatory protein "repressor" coded by *cI* is the one that switches all the other lambda genes off, hence we will concentrate on studying it in this chapter. This situation means that *cI* is the only gene that remains on during lysogeny. It also autoregulates itself to maintain the correct level of the protein under normal lysogenic conditions. It should be noted that P_{RM} needs positive control in order to be able to turn the gene *cI* on, i.e. it needs a protein to bind to the DNA molecule to facilitate the binding of RNA polymerase to the promoter P_{RM} (by recruitment, scenario 3 in table 7-1). This protein is "repressor" hence the autoregulation.

On induction, lytic growth ensues and *cI* has to be switched off to allow the other lambda genes to be expressed. This occurs when the concentration of "repressor" falls and cro starts to take over by first turning *cI* off further decreasing the concentration of "repressor", then turning the other genes on in a specific order. Note that P_R does not need a regulatory protein to help it bind RNA polymerase, as it is a strong promoter.

The operator region that overlaps the two promoters plays a crucial role in this programme. As depicted in figure 7-2, the operator has three sites of equal length and similar nucleotide sequence. Each has a different affinity for "repressor" and for cro, this is important because it determines which site fills first as the "repressor" concentration rises. It should be noted that both "repressor" and cro normally exist as dimers, i.e. two identical molecules (monomers) connected together. Each monomer consists of two domains, an Amino domain that contacts the operator sites on the DNA molecule, and a Carboxyl domain that interacts with another dimer. See figure 7-3, adopted from Ptashne and Gann (2004).

The site with the most intrinsic affinity for repressor is O_R1, then both O_R2 and O_R3 have the same intrinsic affinity. However, because the binding of a "repressor" dimer to O_R1 aids another "repressor" dimer to bind to O_R2 (by cooperativity, scenario 4 in table 7-1), this makes the effective affinity of O_R2 to repressor much higher than that of O_R3.

With cro the situation is the opposite, O_R3 has more affinity for cro than both O_R1 and O_R2 whose affinities are equal. Unlike repressor however, there is no cooperativity between proteins occupying adjacent operator sites with regard to binding cro, because the promoter of its gene (*cro*) is strong. The different affinities are summarised in table 7-2.
So how does this arrangement work in controlling lambda gene expression? We consider this in the next section.

Table 7-2: Affinities of the three relevant operator sites of phage lambda to the regulatory proteins "repressor" and cro. The number of "+" signs indicates the strength of the affinity.

Protein	Affinity to protein			Remark
	O_R3	O_R2	O_R1	
Repressor	+	+ (intrinsic) ++ (effective)	+++	Cooperativity - binding of repressor to O_R1 increases the affinity of O_R2 to repressor
cro	+++	+	+	No cooperativity - binding of cro to O_R3 does not affect the affinity of O_R2 to cro

7.3.2 Operation of the lambda switch

We now examine the operation of the lambda switch by considering the different scenarios that can take place with regard to "repressor", how it binds to the operator sites and the effect of that on the expression of the two genes. We present each scenario briefly as a series of successive steps each leading to the one following it. Figure 7-2 should be referred to at each step to help elucidate the discussion. The different scenarios are summarised in table 7-3 at the end of this section.

1. <u>No "repressor" protein present</u>

- Since there are no "repressor" molecules, which are needed to aid RNA polymerase to bind to P_{RM}, then positive control cannot take place.
- RNA polymerase cannot bind to P_{RM}
- The gene *cI* cannot be switched on.

As an aside, we consider the effect of the lack of "repressor" on the gene *cro*.

- because P_R does not need positive control
- RNA polymerase will bind to P_R
- The gene *cro* will be switched on and will start producing the protein cro
- Because of the high affinity of O_R3 to cro, cro will bind to it first
- Since O_R3 overlaps with P_{RM}, hence the binding of cro to O_R3 ensures that RNA polymerase cannot bind to P_{RM} by exclusion
- This ensures that the gene *cI* cannot be switched on.

Since our purpose here is to illustrate the modelling methodology to follow in later section rather than explain the details of the lambda operation, we will thus limit our discussion to "repressor" and will not follow cro in much detail.

2. <u>Low concentration of "repressor"</u>

Note that "repressor" will mainly be present in the lysogenic state where no cro is present. The first molecules of "repressor" will be synthesised by switching *cI* using a promoter other than P_{RM} which we will not discuss here. Hence we will assume that "repressor" is present.

Case 2.1

- Since O_R1 has the highest affinity for "repressor", then the first "repressor" molecules will bind to it
- Since the O_R1 region overlaps with the promoter P_R, binding "repressor" to O_R1 will prevent RNA polymerase from binding to P_R by exclusion
- This will switch the gene *cro* off, or prevent it from turning on.
- However, because P_{RM} needs positive control to bind RNA polymerase, and since O_R1 is too far from P_{RM} to effect positive control, then RNA polymerase will not bind to P_{RM}
- The gene *cI* cannot be switched on.

Under normal conditions this scenario is not observed because cooperativity between "repressor" dimers ensure that once one dimer binds to O_R1, almost immediately another binds to O_R2 making the two sites fill virtually simultaneously, see case 3.1 below.

For the sake of the modelling exercise that follows, we consider two other cases of low "repressor" concentration; namely what happens when "repressor" binds to either of O_R2 or O_R3 on its own. It has to be emphasised that again, under normal conditions this situation cannot be observed, but it can be set up experimentally by mutation of the operator region. Studying those two situations is instructive in understanding the effects of both operator sites (O_R2 and O_R3) on the molecular interactions between the different players involved in regulation.

Case 2.2

In the case of a mutated lambda chromosome that is missing the operator sites O_R1 and O_R3, at low "repressor" concentration, the following scenario will take place

- "repressor" will bind to O_R2, which will cause positive control,
- thus RNA polymerase will bind to P_{RM}
- hence switching the gene *cI* on

It should be noted that since the O_R2 site overlaps with the promoter P_R, binding "repressor" to O_R2 will prevent RNA polymerase from binding to P_R by exclusion,

hence preventing the gene *cro* from switching on. It should also be noted that because O_R2 is slightly closer to P_R than it is to P_{RM} and because of other molecular mechanisms involved, this negative control by exclusion does not occur to the gene *cI*.

Case 2.3

In the case of a mutated lambda chromosome that is missing the operator sites O_R1 and O_R2, at low "repressor" concentration, the following scenario will take place

- "repressor" will bind to O_R3 which overlaps with P_{RM}
- Hence RNA polymerase cannot bind to P_{RM}
- The gene *cI* cannot be switched on.

However, now because O_R3 is too far from the promoter P_R of the gene *cro*, it will not exercise negative control on it. Hence RNA polymerase will bind to P_R which will switch the gene *cro* on.

3. Medium concentration of "repressor"

We first consider the normal situation then look into the mutated ones.

Case 3.1

- "repressor" will bind to O_R1 switching *cro* off as in the preceding case.
- When a "repressor" dimer binds to O_R1 it helps another "repressor" dimer to bind to O_R2 by cooperativity. This happens almost immediately after the first dimer binds to O_R1.
- Once a "repressor" is bound to O_R2, it facilitates the binding of RNA polymerase to P_{RM} by positive control (figure7-3).
- This will switch the gene *cI* on, which will produce more "repressor" in a positive feedback loop.

Compare this with the switching on of *cI* in case 2.2 where positive control existed but cooperativity did not.

Again we consider two mutated cases that are not observed under normal conditions.

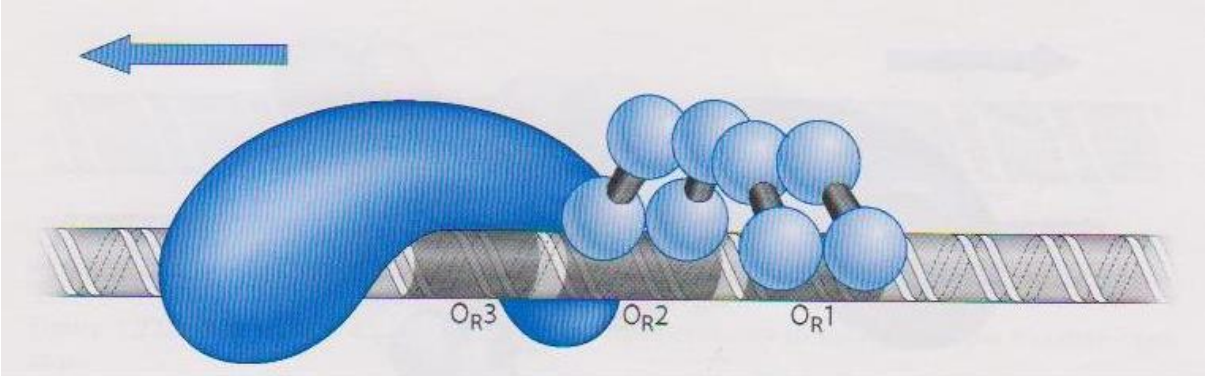


Figure 7-3: Cooperativity between two "repressor" dimers and the recruitment of RNA polymerase. Adopted from Ptashne & Gann (2004)

Each of the dumbbell shapes represents a "repressor" monomer consisting of two domains. Two dumbbells form a dimmer.

Case 3.2

In the case of a mutated lambda chromosome that is missing the operator site O_R1, at medium "repressor" concentration, the following scenario will take place

- "repressor" will bind to O_R2,
- By cooperativity, another "repressor" dimer will bind to O_R3 which overlaps with P_{RM}
- Hence RNA polymerase cannot bind to P_{RM}
- The gene *cI* cannot be switched on.

Case 3.3

In the case of a mutated lambda chromosome that is missing the operator site O_R2, at medium "repressor" concentration "repressor" will bind to both O_R1 and O_R3 excluding RNA polymerase from either of P_R and P_{RM}, thus switching both genes (*cI* and *cro*) off.

4. High concentration of "repressor"

When the concentration of "repressor" becomes high due to some possible problem with the host cell, like stopping division which accumulates "repressor" dimers; then the excess "repressor" dimers will bind to O_R3 leading to the following scenario

- "repressor" binds to O_R3
- Since O_R3 overlaps with P_{RM}, hence the binding of "repressor" to O_R3 prevents RNA polymerase from binding to P_{RM} by exclusion
- This will switch the gene *cI* off

This will prevent further synthesis of "repressor" reducing its concentration in a negative feedback loop.

In summary, whenever a regulatory protein binds to an operator site that overlaps with the promoter of a gene, it prevents RNA polymerase from binding to that promoter by exclusion hence preventing transcription of the gene. For the two lambda genes *cI* and *cro*, this can be stated as follows

- Whenever a regulatory protein - be it "repressor" or cro - binds to the operator site O_R3, the gene *cI* will be switched off.
- Whenever a regulatory protein - be it "repressor" or cro - binds to either of the operator sites O_R1 or O_R2 (or both), the gene *cro* will be switched off.

The different scenarios explained above for the effect of the concentration of the regulatory protein "repressor" on the expression of the genes *cI* and *cro* is summarised in table 7-3. A "0" in the column of an operator site indicates that the site is not occupied, while a "1" indicates that it is occupied by the regulatory protein "repressor". The first column in the table refers to the case number of the different cases discussed in the text above. Note from the table that the two genes cannot be on together, although they can be both off at the same time.

Table 7-3: Effect of "repressor" concentration on the state of genes *cI* and *cro*. (Case number refers to the numbers in the text).

Case number	O_R3	O_R2	O_R1	Gene *cI*	Reason for gene *cI* switch off	Gene *cro*
1	0	0	0	Off	No +ve control	On
2.1	0	0	1	Off	No +ve control	Off
2.2	0	1	0	On	-	Off
3.1	0	1	1	On	-	Off
2.3	1	0	0	Off	-ve control (exclusion)	On
3.3	1	0	1	Off	-ve control (exclusion)	Off
3.2	1	1	0	Off	-ve control (exclusion)	Off
4	1	1	1	Off	-ve control (exclusion)	Off

Note that only three cases, namely 1, 3.1 and 4 are normally observed; those are the ones directly related to "repressor" concentration. The first is observed in the absence of repressor, the second at low "repressor" concentration and the third at high concentration. This reduces table 7-3 to table 7-4 below.

Table 7-4: Observable states of the genes *cI* and *cro* at different "repressor" concentrations.

Repressor concentration	O_R3	O_R2	O_R1	Gene *cI*	Remark on gene *cI*	Gene *cro*
None	0	0	0	Off	No +ve control	On
Low	0	1	1	On	cooperativity	Off
High	1	1	1	Off	-ve control	Off

Following a similar reasoning as for the case of the regulatory protein "repressor", and noting that cro is not involved in cooperativity or positive control (recruitment), the different scenarios for the effect of the concentration of the regulatory protein cro on the expression of the genes *cI* and *cro* can be worked out. The results are similar to those in table 7-3 for the gene *cro*, although slightly different for the gene *cI*. As in the previous discussion, some of the cases are not observed under normal conditions and can only be devised experimentally. The different observable scenarios for the gene *cro* are summarised in table 7-5, where now four different concentrations of the protein cro are considered. The lack of cooperativity between cro dimers means that single dimer binding can take place, unlike "repressor" where one dimer immediately recruits another rendering single dimer binding unobservable under normal conditions. Also lack of cooperativity means that at medium cro concentration, two dimers do not need to bind to adjacent operator sites.

Table 7-5: Observable states of the gene *cro* at different concentrations of the protein cro.

cro concentration	O_R3	O_R2	O_R1	Gene *cro*	Remark on gene *cro*
None	0	0	0	On	No -ve control
Low	1	0	0	On	Affinity of O_R3 is highest
Medium	1	1	0	Off	-ve control (exclusion)
Medium	1	0	1	Off	-ve control (exclusion)
High	1	1	1	Off	-ve control (exclusion)

We now have enough details about the construction and operation of the phage lambda system that allow us to model it. Further information on phage lambda and on the different molecular interactions involved in gene regulation can be found in the

two books by Ptashne and Gann (2002, 2004). It is important not to lose track of the actual purpose of this chapter and indeed of the whole of this work, and that is to introduce a modelling method rather than produce a particular model. Since methods for modelling the regulation of gene expression have already been reviewed in chapter three we will only mention here how they have been applied to phage lambda, without going into the mathematical details of the resulting models. Arguably one of the earliest mathematical models for phage lambda was that by Ackers *et al.* (1982) and it used statistical thermodynamics to determine the probabilities of binding of the "repressor" and cro proteins to the different operator sites, in essence determining the affinities; the model used differential equations. This model was later expanded upon by Shea and Ackers (1985) and by Santillan and Mackey (2004b). As has been discussed in chapter three, the major problems with such models include their complexity, uncertainty of the molecular mechanisms involved and the large number of unknown parameters, most of which have to be estimated from the data or assumed. Another modelling approach used is the discrete one, and here because of the multiple-valued nature of the proteins' concentrations, generalised Boolean networks were used (Thieffry and Thomas 1995), those were also discussed in chapter three and their drawbacks were pointed out, in particular the awkward and non-intuitive mathematical formulation resulting.

We reiterate that the interest is in the modelling methods and not in the systems being modelled, thus in the context of this work, phage lambda is just a vehicle for delivering the method.

7.4 A binary model for gene regulation in phage lambda

It is clear from table 7-3 that the gene regulatory function of phage lambda lends itself readily to binary models. The usual practice in applying this modelling approach is to consider the binary values to indicate the crossing of some concentration or activation threshold by some variable, hence taking a functional view. However, the particular case of phage lambda as summarised in table 7-3 indicates that we can also take a structural view of the situation in the sense that the binary values can indicate the presence or absence of a molecule at a certain site. The effect on the gene can still be considered in the functional view, i.e. that it is switched

on or off. With this in mind, table 7-3 can be reformulated into table 7-6, which now represents a logic function where we have associated a binary variable with the state of each operator site, and with the expression states of the genes.

Table 7-6: A binary representation of the functions in table 7-3.

Minterm number	O_R3 (x_3)	O_R2 (x_2)	O_R1 (x_1)	Gene *cI* (y_1)	Gene *cro* (y_2)
0	0	0	0	0	1
1	0	0	1	0	0
2	0	1	0	1	0
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	0	0

Recall from the previous chapter that the Reed-Muller expansion of a three variable binary logic function y is given by

$$y = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_2 x_1 \oplus a_4 x_3 \oplus a_5 x_3 x_1 \oplus a_6 x_3 x_2 \oplus a_7 x_3 x_2 x_1 \qquad 7\text{-}1$$

Where the coefficients a_i are obtained from the function values d_i by the transformation

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} \qquad 7\text{-}2$$

From table 7-6 the two functions describing the states of the genes *cI* and *cro* are given by the vectors below representing the functions' values (the d_i's), where the prime sign indicates the transpose of a vector.

$$y_1' = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$y_2' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Substituting in equation 7-2 and multiplying the vector by the matrix in GF(2), we get the coefficient vectors below

$$a_{y_1}' = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]$$
$$a_{y_2}' = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$$

Substituting in equation 7-1 we get the RM expression for both functions

$$y_1 = x_2 \oplus x_3 x_2 \qquad 7\text{-}3$$

$$y_2 = 1 \oplus x_1 \oplus x_2 \oplus x_2 x_1 \qquad 7\text{-}4$$

Let us examine the gene regulatory function y_1 in equation 7-3 modelling the response of the gene *cI* to the different occupation states of the operator sites O_R1, O_R2 and O_R3, by the "repressor" protein. We can make the following observations (see also figure 7-2):

- The constant term (a_0 in equation 7-1) is zero indicating that the basal state of the gene is off, i.e. in the absence of any regulating conditions the gene is off. Such conditions would then modulate it in such a way as to either switch it on or keep it off.
- The state of the gene does not depend on x_1, which makes biological sense since as mentioned in the different scenarios above, O_R1 is too far from the promoter P_{RM} of the gene to effect any control (positive or negative) on it.
- x_2 is necessary but not sufficient to switch the gene on.
  - The necessary part: Mathematically, this means that we cannot have $y_1 = 1$ unless we have $x_2 = 1$. Biologically this means that the gene cannot be switched on if O_R2 is not occupied by "repressor", which makes sense because of the positive control exerted by the "repressor" protein when it occupies the operator site O_R2.
  - The sufficiency part: Mathematically having $x_2 = 1$ is not a sufficient condition to have $y_1 = 1$, since we can have $x_2 = 1$ but still get $y_1 = 0$, and that is when we simultaneously have $x_3 = 1$. Biologically this means that having positive control is not sufficient to turn the gene on if we have negative control at the same time (by exclusion when O_R3 is occupied).
- x_3 on its own is sufficient to switch the gene off, but it is not necessary. Mathematically, this means that if we have $x_3 = 1$ in equation 7-3, then

irrespective of the value of x_2, we will get $y_1 = 0$ indicating that the gene will be switched off. Biologically this means that the gene cannot be switched on if O_R3 is occupied by "repressor" irrespective of whether O_R2 is occupied or not, because of the negative control exerted by the "repressor" protein when it occupies the operator site O_R3 excluding RNA polymerase. A similar argument as in the point above demonstrates that x_3 is not necessary for switching the gene off, since it can be switched off in the absence of "repressor" binding to O_R3, if it is also not bound to O_R2.

The two last points above can be summarised as follows:

1. x_2 is necessary but not sufficient to switch the gene on, i.e. the gene cannot be switched on without x_2, but x_2 on its own is not sufficient to switch it on as we also need to ensure that x_3 is not present. Biologically this means that positive control is necessary to switch the gene on, but it is not sufficient as we need to ensure that there is no negative control. In other words, if both controls are present then negative control is dominant over positive control.
2. x_3 is sufficient but not necessary to switch the gene off, i.e. x_3 on its own is enough to switch the gene off, but if it is not present then the switch can still be switched off by lack of x_2. Biologically this means that negative control is sufficient to switch the gene off (because of exclusion of RNA polymerase), but it is not necessary, i.e. even if there is no negative control the gene can still be switched off if there is no positive control (because RNA polymerase will not be able to bind to the promoter site even if it is accessible).

Similarly we can examine y_2 in equation 7-4, which models the response of the gene *cro* to the different occupations of the operator sites O_R1, O_R2 and O_R3 by the "repressor" protein. We can make the following observations

- The constant term (a_0 in equation 7-1) is now 1 indicating that the basal state of the gene is on, i.e. in the absence of any regulating conditions the gene is expressed. Such conditions would then modulate it in such a way as to either switch it off or keep it on.
- The state of the gene does not depend on x_3, which makes biological sense since O_R3 is the only one of the operator sites that does not overlap with the promoter P_R of the gene, and hence cannot effect negative control (by

exclusion) on it. Note that unlike the gene *cI*, the gene *cro* does not need positive control to be switched on; hence the lack of positive control does not switch it off.

- Any one of x_1 or x_2 on its own is sufficient to switch the gene off, and if they both are = 1, then this will also switch the gene off. So effectively the outcome of both variables is an OR gate sort of response. This makes sense biologically, because if either (or both) operator site O_R1 and O_R2 is occupied it will exert negative control (by exclusion of RNA polymerase) on the promoter P_R of the gene. Mathematically, if any or both of the variables in equation 7-4 is 1 we get $y_2 = 0$ indicating that the gene will be switched off, which means that it is necessary to have at least one of them present.

The discussion above demonstrates the power of combining a well grounded biological understanding with a mathematically powerful modelling technique. It should be noted that other Boolean models such as the Disjunctive Normal Form (DNF) will lead to the same results but they lack the elegant interpretive power of the Reed-Muller formulation as demonstrated above. In addition the RM formulation can be extended to the multiple-valued case in a straightforward manner as demonstrated in the next section, hence providing a wider modelling scope.

7.5 A multiple-valued model for gene regulation in phage lambda

It is clear from table 7-4 above relating the concentration of the protein "repressor" to the states of the genes, that there are three concentration thresholds that are significant in determining the state of the genes, whether on or off. Table 7-4 is repeated below in a condensed form as table 7-7 in which the status of the operator sites has been removed as they are not relevant to the modelling exercise to follow.

Details of the multiple-valued modelling approach have been presented in the previous chapter. The first step in applying it is to choose an appropriate set on which to define the functions. It is clear from table 7-7 that there are three significant values for the input variable, qualitatively termed None, Low and High, hence a suitable set

would be {0, 1, 2}. It should be made clear that the values in the set do not represent their numerical values but rather certain thresholds for the concentration. In other words, the value 1 does not mean 1 mol/unit volume or any other unit of concentration, but means that it is above the first concentration threshold, similarly for the other elements of the set. It should further be noted that the differences between the concentration thresholds need not be equal, i.e. the range from 0 to 1 is not necessarily the same as from 1 to 2. This is the consequence of the abstraction process.

Table 7-7: Effect of "repressor" concentration on the states of the genes *cI* and *cro*, as presented in more detail in table 7-4 above.

Repressor concentration (x)	**Gene *cI* (y_1)**	**Gene *cro* (y_2)**
None	Off	On
Low	On	Off
High	Off	Off

The situation of the output is not as clear as it is for the input; granted it only takes two values but now we have two options. The first is to define the output variables to belong to the binary set {0, 1} which is intuitive but will complicate the mathematics. The second is to define it on the same set as the input (which has three elements) but restrict the values to only two of those three elements. We will choose the second option, but this raises another question, which two values of the three available do we pick? A common approach is to use the extremes of the set, i.e. the values 0 and 2.

The second option leads to a simpler mathematical formulation, since now both the input and outputs are defined on the same set. This means that we can define the functions on a finite field and use the formulations presented in the previous chapter. In this particular case, the functions can be defined on the finite field GF(3), and represented as in table 7-8.

Table 7-8: The functions of table 7-7 represented on the finite field GF(3).

Repressor concentration (x)	**Gene *cI* (y_1)**	**Gene *cro* (y_2)**
0	0	2
1	2	0
2	0	0

A single variable ternary function will have the form

$$y = a_0 + a_1 x + a_2 x^2 \qquad 7\text{-}5$$

Where the coefficients a_i are obtained from the function values d_i by the transformation

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} \qquad 7\text{-}6$$

and all computations are performed using GF(3) arithmetic, i.e. modulo 3, which leads to the two functions

$$y_1 = x \oplus x^2 \qquad 7\text{-}7$$

$$y_2 = 2 \oplus x^2 \qquad 7\text{-}8$$

In an interpretation similar to that in the previous section, it is clear from substituting the different values for x in equations 7-7 and 7-8 that the gene *cI* is normally off (because a_0 of equation 7-5 is zero), and that it is switched on at a low concentration of "repressor" and switched back off at high concentration. On the other hand the gene *cro* is normally on (because a_0 of equation 7-5 is not zero), and is switched off by any presence of repressor whether at a low or a high concentration.

To be able to infer and predict the behaviour of the regulatory function using the multiple-valued models, we need to actually substitute values in the equations, unlike the binary case where prediction is made simply by inspection. This is because of the somewhat counterintuitive nature of non-binary finite fields, where x^2 is not always greater than x but depends on the value substituted (e.g. on GF(3) for $x = 2$ we have $x^2 = 1$). Note that a different choice of values for the gene expression levels, say 0 and 1 instead of 0 and 2, will have resulted in a function y_1 of a different form than that of equation 7-7, but of the same values for the different inputs; similarly for y_2.

In the above modelling exercises we have studied the effect of the concentration of the protein "repressor" on the two genes *cI* and *cro*, the same can be followed for the protein cro. Both a binary and a multiple-valued model can be developed in a similar

manner, and with a corresponding interpretation of the results. It should be noted that for the multiple-valued case, examination of table 7-5 above indicates that there are four concentration thresholds of the protein cro that are significant in determining the state of the gene *cro*. Hence this can be considered as a function on the finite field GF(4), where now the relevant arithmetic operations (see chapter six) and transformation matrices have to be used (Green and Taylor 1974).

In the multiple-valued models for phage lambda developed above, whilst the regulatory inputs were multiple-valued, gene expressions were binary, i.e. either expressed or not. There are situations however, in which the gene expression levels are multiple-valued in nature, such as the case with morphogenesis as explained in chapter two. Indeed, we have modelled a regulatory process in the formation of the sense organs in the fruit fly *Drosophila* described by Ghysen and Thomas (2003). The process involves a gene controlled by two regulatory signals that take ternary values, and that lead to three expression levels for the gene, each triggering a different response in the cell. Our model is described in detail elsewhere (Aleem *et al.* 2008), see Appendix II and demonstrates two features of the multiple-valued model that are not present in the phage lambda case. Firstly the output can also be multiple-valued, not just the input, and secondly a case of two multiple-valued inputs.

7.6 Conceptual differences between the binary and multiple-valued models

We have produced two models for the phage lambda system above, namely a binary model and a multiple-valued one. It is fitting at this point to make a conceptual comparison between the two. It is obvious that the two models are mathematically different, however more important than the difference in the mathematical details, is the fact that they reflect different modelling perspectives.

In the particular models above, the binary case provides what can be termed a mechanistic model, in the sense that it is developed based on an understanding of the molecular mechanisms involved in regulating the genes. These include the order of binding of the regulatory protein to the different operator sites and the effect that has

on the expression of the genes. On the other hand, the multiple-valued model provides what can be considered a phenomenological model, in the sense that it is based on external observations (of the phenomena) without necessarily any knowledge of the intricate details of how those phenomena take place on a molecular level. That is because the observations are made of the different protein concentrations and the corresponding gene expression levels, without predetermined knowledge of how one affects the other. We stress that this is the situation in this particular case and that in general either approach can be used to describe either a mechanistic situation or a phenomenological one. In other words a binary approach may very well be used to build a phenomenological model in a different context, and the multiple-valued approach used for a mechanistic model in another. Thus the choice is not about which mathematical tool is used, but what sort and amount of information is available about the system to be modelled. This ties back neatly with the discussion of the conceptual issues related to modelling in chapter three, in particular those concerning modelling decisions.

7.7 Discussion

In chapter five we have introduced our modelling method for binary gene regulatory functions, based on the Reed-Muller expansion, and demonstrated some innovative interpretations and applications for it. This was continued in chapter six where we then extended it to the multiple-valued case. In this chapter, chapter seven, we applied the method to model a gene regulatory system from the literature. We can step back now from the technical details involved, mathematical and biological, look at the big picture and take stock of the situation, discuss what we have achieved, its advantages and disadvantages, and what future directions of research it can spur.

We have developed a method for modelling gene regulatory functions for which only a limited number of discrete values are of interest. The more common binary case is a special case of this method. The method can be thought of as semi-qualitative, in the sense that it gives more than just the on/off information of the Boolean formalism, although not the fully quantitative results of the differential equations formalism. Hence compared with the Boolean method it provides more information about the system and allows building models from more than one perspective, as in the case of

the phage lambda above. Also compared to the other multiple-valued method, known as generalised Boolean networks, our method is more intuitive, computationally easier and makes more sense analytically because all the variables represent actual regulatory factors and not meaningless dummy variables.

In terms of the disadvantages of the method, the main issue is the situation when not all the variables have the same number of values, i.e. they belong to different sets. In this case one is forced to use the set with the largest number of values (largest cardinality), which may make the resulting model more complicated than is necessary. This however, is not as serious as it may sound because determining the coefficients which define the model is a straightforward matrix multiplication operation, albeit on a finite field. Commercially available mathematical software packages do implement operations on finite fields, in particular MATLAB and Mathematica. Another difficulty when implementing our method lies in the optimisation of the resulting expression, and by that we mean mathematically manipulating it to obtain the expression with the least number of terms. There are techniques for doing that, and they involve such approaches as allowing complementation of some of the variables; recall that the Reed-Muller expansion in its original form is complement free. Optimisation however, is more important in synthesis than in modelling.

So what further avenues of research does our approach open up? There are quite a few, perhaps the most beneficial one in the context of gene regulation is to model dynamic processes, or sequential networks as they are known in electronic engineering. The work of Laubenbacher deserves special mention in this respect, as he and co-workers have addressed this problem, although from a mathematical point of view (Laubenbacher and Stigler 2004; Jarrah *et al.* 2007). Their work however, is highly abstract and is very difficult to follow; we believe that our approach can be extended to the dynamical case in a more intuitive way that relies on an engineering rather than a mathematical approach. Another area in which our method can be applied is in multi output functions, i.e. the situation where the same inputs affect several outputs at the same time. This is similar to the case of phage lambda above where the same protein affects two genes. The idea in such a modelling approach is to treat the two outputs as one output on a larger finite field. For example two binary outputs, such as *cI* and *cro*, above can be treated as one output on GF(4), but now this

output will not have direct physical significance and any results obtained will have to be translated back to the original outputs. So what would be the benefit of such a modelling approach? It is to study both outputs together rather than separate functions, a situation that can be very helpful in drug design for example, to study the effect of a drug, which is essentially a regulatory factor, on more than one process at the same time. This would be beneficial in studying side effects of a drug.

Another area that we believe can be very promising in analysing gene regulatory functions is to extend the transform approach to orthogonal transforms such as the Walsh transform. This we believe, would allow studying certain classes of Boolean functions that are becoming increasingly important in biological applications, and are known as canalysing functions, where under certain conditions one input dominates all the others (Kauffman 1993; Kauffman *et al.* 2004; Reichhardt and Bassler 2007). Finally the method can be applied in different contexts such as in synthetic biology, other regulatory functions, or other biological or non-biological contexts in general.

7.8 Summary and conclusion

This chapter covered three main issues in a particular order reflecting a logical progression of ideas whereby each furnishes a basis for the one to follow.

We started by a review of the molecular interactions between proteins (including regulatory proteins and RNA polymerase) and DNA sites (including promoters and operators), and between proteins and each other. Those interactions form the basis of such concepts as positive and negative controls, and cooperativity all of which affect the regulation of a gene. Understanding those interactions and the resulting regulatory effects is important in understanding transcription regulation in any system; hence the coverage in this part of the report was generic and not related to any particular system (organism).

Next we introduced phage lambda, its structure and function and how it regulates it. Here we used the material in the first part, about regulatory mechanisms, to explain how phage lambda effects its regulation. Phage lambda was not introduced for its own sake but as an instance of a gene regulatory process that is well studied in the literature.

In the third part we used the detailed explanation of the regulation of the phage lambda genes as a case for applying our modelling approach developed in the previous chapters. We produced two models a binary and a multiple-valued one.
We have looked at the binary Reed-Muller expansion of phage lambda in some detail where the interpretive power of this approach has been demonstrated by neatly tying the mathematical aspect of the model to its biological interpretation. A different modelling perspective based on the multiple-valued Reed-Muller expansion was also demonstrated and briefly discussed. A conceptual comparison between the two models was made.

This chapter is the culmination of this work; it brings together concepts from the previous chapters and weaves them together into a single integrated fabric. The discussion of the molecular interactions involved in gene regulation builds on the material in chapter two concerning gene expression and its regulation. The models built here also use the methods developed in chapters five and six on the Reed-Muller expansion for the binary and multiple-valued case. But perhaps most importantly, on an intellectual level this chapter ties well with the material at the beginning of chapter three concerning the conceptual issues underlying modelling. It demonstrates the abstract issues discussed there in a concrete way by applying them to an actual biological system. Such issues include the importance of understanding the underlying biological processes in model building, and how the same data can produce different conceptual models; e.g. structural vs functional, or mechanistic vs phenomenological, and how the different model building decisions affect the resulting model.

We concluded this chapter with a discussion of the merits and drawbacks of our method and the potential future research.

Chapter 8: Summary, Conclusion and Future Research

In this chapter I will summarise the work in this report, point out to its contribution and the possible areas of application. I will close with a discussion of how this work can be taken forward both in the context of the regulation of gene expression and in a wider context. I will present the material here as a series of questions, in a logical order whereby each question leads to the one that follows. Next to each question, in parentheses, is the topic the question relates to. In answering those questions I will refer to the relevant sections in the body of this work. I believe this way of presenting the material will make the conclusion more concise and focus the attention on the pertinent issues.

I will imagine I am being asked these questions by the reader in the form of a conversation, hence the presentation will be somewhat informal, which is why I am using the first person in this chapter.

1. **What is the purpose of this work?** (Aims)

The purpose of this work is to develop a method for the mathematical modelling of the regulation of gene expression that can accommodate multiple-valued discrete expression levels. See chapter 1, section 1.3.

2. **Why is this a problem worth investigating?** (Motivation)

We can think of the motivation for this work to be at two levels, motivation for the method and motivation for the problem the method is investigating. The problem we are ultimately helping to investigate is that of the regulation of gene expression.

Motivation for the problem: understanding the regulation of gene expression is fundamental to understanding many of the biological processes inside the cell and ultimately the organism. For life scientists this helps in understanding both the normal and some of the anomalous situations in nature such as diseases (e.g. cancer and genetic disorders) and other biological phenomena. Attaining such understanding would help to cure or prevent diseases or manipulate the underlying biological controls in general. For us engineers, understanding such regulatory activities inside the cell can help us optimise process design and improve process troubleshooting in the biotechnology industry. See chapter 1, section 1.1.

Motivation for the method: The proliferation in the quantity and quality of information related to the regulation of gene expression collectively known as omics was a boost for the use of mathematical modelling of such regulatory processes. Quantitative models suffer from the uncertainty in deciding on the molecules and mechanisms involved in the regulation and the values of the parameters used in the models. Qualitative models, especially those based on Boolean algebra give a simple and intuitive approximation but they fail to capture the case where there are multiple discrete values for the different biological variables. This is where our method comes in. See chapters 2 and 3.

3. If this is such an important problem, surely others must have attempted to solve it? (Literature survey)

Yes of course, there is an extensive literature on mathematical modelling of the regulation of gene expression. In fact there are even whole journals exclusively dedicated to mathematical, theoretical and systems biology, all of which more or less deal with modelling of biological processes in general, including the regulation of gene expression. Models in the literature can be broadly classified into those with continuous variables and those with discrete variables. The variables involved include expression levels of genes and concentrations of different regulatory molecules.

The most common continuous models are based on ordinary linear differential equations, less common ones use non-linear and partial differential equations.

The most common discrete models are based on Boolean algebra which assumes binary values for the discrete variables. Other formalisms for discrete variables also exist.

The accuracy of the analysis and prediction provided by quantitative models is undermined by the uncertainties involved in building those models, such as which molecules are involved in the regulation of gene expression, the mechanisms by which they are involved and the values of the different parameters in the model. This necessitated the use of qualitative models, which are based on the assumption that the different biological values exist in only one of two extreme states, leading to the use of the Boolean formalism to build the model. Whilst simple and intuitive, not all gene regulatory functions fit in this view, for example in morphogenesis there can be multiple threshold levels for the biological variables.

There are other classifications of modelling approaches such as dynamic *vs.* static and deterministic *vs.* stochastic, with examples in the literature. See chapter 3, sections 3.5 to 3.7.

4. What is the method you are using? (Method)

The method I am using to model multiple-valued discrete regulatory functions is based on finite fields algebra. The core of the method is that any function defined on a finite field can be represented as a polynomial on the field with degree less than the order of the field. Finite fields are also known as Galois fields and denoted by GF(q) where q is the number of elements in the underlying set and is known as the order of the field. The trick is then what biological variables to identify with the elements of the field, how to do that, and how to formulate the gene regulatory function based on that.

In the binary case the finite field reduces to the two element field GF(2) in which case the resulting polynomial is known as the Reed-Muller expansion of the function. The multiple-valued case is also known in the engineering literature as the multiple-valued Reed-Muller expansion. See chapters 5 and 6.

5. What are the pros and cons of this method? (Critique)

I like to differentiate between the modelling method (which is based on finite fields algebra), and the methodology used to develop the method. A methodology can be thought of as a meta-method, i.e. a method or process for developing methods. My methodology comprises three stages. Firstly a process of abstraction; secondly is mathematical modelling as a tool in general, and thirdly the particular method. Let us now look at the pros and cons of each of these levels.

a. Abstraction

Advantages: Allows for the simplification of the system being considered by detaching it from its implementation or domain specific details. This allows for identifying the commonalities between the problem at hand and similar ones in other domains of knowledge, hence allowing the use of methods already tried and tested in those other domains.

Disadvantages: Inevitably and by definition it loses the details of the issue being investigated, but this is not a problem as the core issues which are the target of the investigation are retained. Furthermore, it requires a high degree of abstract thinking, possibly more than is normally employed in engineering. See chapter 1 and chapter 3, sections 3.1 to 3.4.

b. Mathematical modelling as a tool

Advantages: It assimilates and integrates large amounts of data or observations to give a concise description of the system or process being modelled. This allows for mathematical manipulations and derivations that may give different insights into the problem, and that might not be obvious from the data. It can also be used to predict un-tested cases, generate hypotheses and run what-if analyses.

Disadvantages: Mathematical modelling is a process with inputs as observed data and existing knowledge about the system, and whose output is the model. The resulting model is determined by the inputs (observations and knowledge) and the mathematical formalism utilised. The observations and knowledge are limited by the accuracy of the observation mechanisms, and the mathematical derivation is

constrained by the validity and scope of the formalism. Those factors limit the validity of the model on the one hand but allow developing several models of the same system on the other; hence allowing different descriptions of the system depending on what is being investigated. See chapter 3, sections 3.1 to 3.4.

c. Our method

Advantages: It allows for modelling of multiple-valued discrete state regulatory systems in a way that is straightforward both in its theoretical background and its practical application.

The theoretical background is abstract algebra, in particular finite fields and function spaces on them. Those are merely abstractions of other algebraic structures very familiar to engineers, namely the real numbers field, and Taylor series and Fourier series which are expansions on appropriate function spaces defined on the real numbers field. See chapter 4.

The application is also straightforward whether in analysis or synthesis of regulatory systems as it merely involves matrix multiplication, albeit on a finite field. The process is completely transparent to the user in the sense that the user need not understand the underlying mathematics in order to use it. This is similar to the colloquial saying "you don't need to understand how a mobile phone works in order to use it!". Furthermore, some of the common mathematical software packages implement finite fields arithmetic. See chapters 5 and 6.

Disadvantages: The mathematics might appear a bit obscure and counterintuitive to some, but we have clarified above that this is because of its abstract nature. Also there is a difficulty in applying the multiple-valued case when not all the variables have the same number of values, for example when some are ternary and others are binary, as in the phenomenological model of phage lambda. Thus choices have to be made regarding the mathematical values to assign to the biological variables, and even though these may lead to different mathematical forms for a given function, the biological input/output relationship will not change. Also the multiple-valued case is limited to discrete variables whose number of values is a prime or a positive integer power of a prime. This should not be a problem however, since of all the numbers

from 1 to 9, only the number 6 does not belong to this category. Variables with more than 9 values will lead to exceedingly complicated functions even if they fall in this category. In spite of its disadvantages it remains more intuitive than similar methods in the literature. See chapter 6, section 6.7 and chapter 7, section 7.7.

6. **How does your approach differ from that of others?** (Literature survey)

See questions 3 and 5 above.

7. **So what is the outcome of your work?** (Results)

The outcome of my work is threefold

- A method for modelling multiple-valued discrete gene regulatory functions, together with an elegant biological interpretation of it.
- Three different mathematical interpretations of this method namely: a function on a Boolean algebra, a polynomial on a finite field and a transform on a discrete function space.
- Three possible biological applications of these mathematical interpretations, respectively reverse engineering of gene regulatory functions, detection of mutations that affect the function of a protein, and synthesising biological regulatory functions.

Therefore the outcome is both theoretical and applied. It is important to note though that this work is about developing a method not producing a model, hence the systems used to demonstrate the validity of the method were ones that are simple and well understood. Realising the rather abstract nature of the work and the possibility of losing the big picture in the details (the proverbial forest and trees), I decided to use simple systems to avoid masking the method by the fine details of the application.

8. Is this stuff applicable or is it only of theoretical interest? (Critique)

Yes this stuff is applicable and we have demonstrated this in the answer to the questions above. We have also demonstrated it in the main body of the work by modelling phage lambda, and by an example of synthetic biology.

However, the drive for applicability should not take away from the theoretical aspect of the work, since understanding the underlying theoretical issues helps guide the application; e.g. which areas it is likely to be useful in, and what are the limitations of the resulting application. Note that by emphasising understanding of the theoretical issues we do not mean the actual mathematical mechanics (see question 5 above), but rather the conceptual issues involved.

For example when we use the method for modelling, understanding both its conceptual background and the details of the biological problem being modelled will help us decide whether to use a phenomenological model or a mechanistic one. Similarly, when using it for mutation detection, the theoretical understanding will help us decide which types of mutations can be detected, and that is by making the correct analogy with circuit faults. So those are application related issues that are guided by the theoretical ones.

9. What is your contribution? (Contribution)

The contributions can be regarded, somewhat in analogy with question 5, as to be at three levels namely conceptual, mathematical and applied.

Conceptual: Using abstraction I establish an analogy between some problems in biology with others in electronic circuits. Whilst the analogy between the two fields in general is not new, the analogy between the specific problems is to the best of my knowledge novel. Those specific problems are the correspondence between mutations and faults in a circuit, and between linear transformations in engineering and in biology. See chapters 5 and 6.

Mathematical: The use of the Reed-Muller approach in the context of gene regulatory functions and in particular its interpretation as elaborated upon in chapter 5. See chapters 5 and 6.

Applied: The formulation of the biological problems below in the form given in the main body is novel namely:

- The use of exclusive OR operator for reverse engineering studies, in particular pointing to its superior discriminating ability compared to the common logic OR.
- The use of the Boolean difference in mutation detection.
- The transform approach whether for the analysis or synthesis of biological regulatory functions. Also the intuitive conceptual interpretation of a transform and its relation to reversibility in the information theoretic sense. See chapters 5 and 6.

10. How can this work be taken forward? (Future work)

I believe that this work opens up several avenues of research that can be pursued both within the context of this work and in a wider context. I classify those as fundamental and applied, where fundamental here refers to the method and applied to potential applications of the method. See chapter 7 section 7.7.

Fundamental research can be pursued in several directions, for example

- Computational, such as how to formulate faster and more efficient algorithms for computing the Reed-Muller coefficients, especially for the multiple-valued case.
- Analytical, such as how to extend the method to the sequential case where the process would be dynamic, i.e. includes a time element, and how to give an intuitive interpretation to that. Similar work does exist in the dynamical systems literature but is very abstract, tying it to a more familiar and concrete area such as logic design would help clarify it and possibly obtain new results. Similarly the method can be extended to the multi-output case.

- Other analytical research can look into design techniques using the Reed-Muller formulation, in particular the optimisation of the resulting design such as reducing the number of terms in the expansion.

Applied research can also be pursued in several directions for example

- Apply the method to other discrete regulatory processes in the cell or in biology in general, or indeed in any other context.
- Extend the transform method to the orthogonal case. Orthogonal transforms on discrete function spaces do exist, such as the Walsh and Haar transforms but have not been applied in the context of gene regulatory functions. It would be interesting to see what interpretations emerge from such a view and what new problems can be attacked.
- Apply the sequential view mentioned above to the reverse engineering of discrete gene regulatory networks.
- Apply the multi-output formulation to gene regulatory networks. Multi-output refers to the case when the same inputs affect several outputs at the same time. This can be very interesting and valuable in drug design where it can be used to investigate how the effect of a drug which in this context is a regulatory input, affects not only the intended target but other regulatory processes in the cell as well.
- The method can also be used in what-if analyses, to simulate the knockout or silencing of genes and determining what the output will be. This would be the opposite of the mutation problem mentioned above, where now the mutation is intentional and the output function is to be determined.
- Application in synthetic biology such as determining the optimal design and the mapping of the finite field elements to the appropriate biomolecules. Related applications to that can be in modelling the genetic code, as presented in a paper we have published.

In general, application of multiple-valued logic in biological processes might serve to revive this area of research which was highly active in the mid seventies to mid eighties in the electronic design domain. This activity has dampened greatly since

then because of the lack of a "killer application", which may very well be provided by biology, in particular biomolecular computing and synthetic biology.

Final thoughts

I do recognise that this work is rather abstract compared to engineering work at a similar level, however this is the nature of theoretical biology to which this work belongs. Because of that, the presentation is somewhat different from the standard format, and that is why I have included the diagram in chapter one mapping the sections of this work to the corresponding ones in the "standard model" of a thesis. In fact in some parts of this work the presentation may even appear somewhat unorthodox, as in this chapter.

An even more abstract view of this work would regard the methodology as comprising two aspects: analysis and synthesis, essentially deconstructing or dismantling the problem in one domain, carry it over across domain boundaries, reconstruct it or reassemble it in a different domain and see it in a different light and from a different perspective. Indeed, this allows one to ask new questions. In this sense and on a fundamental level, one can consider that the outcome of this work is effectively to help pose different questions rather than provide answers to existing ones. Consequently, I would like to conclude this work with a quotation from a book by Stuart Kauffman, who has worked extensively on Boolean networks and covers them in part of that book. The book is entitled "The Origin of Order: Self-Organization and Selection in Evolution" (Kauffman 1993); in the preface, when discussing how his research interest in the topic developed, he states :

"*The greater mystery, after all, is not the answers that scientists contrive, but the questions they are driven to pose.*"

Appendix I: Published Paper 1

This appendix contains a paper published in the proceedings of the 39th International Symposium on Multiple-Valued Logic, 2009 (ISMVL '09), held in Okinawa, Japan 21-23 May 2009, and organised by the Institute of Electrical and Electronic Engineers Inc. (IEEE), (Aleem *et al.* 2009).
(Note that the page numbers have been changed from the original source to match this report)

The paper applies the multiple-valued Reed-Muller expansion to modelling the genetic code as a function on a finite field. It is included here as it demonstrates how the elements of the field GF(4) can be identified with biomolecules, in this case the four nucleotides A, T, C and G. This relates to the material on synthetic biology in chapter six.

Representing the Genetic Code as a Function on a Galois Field Using the Reed-Muller Expansion

H. A. Aleem
School of Chemical Engineering & Analytical Science, University of Manchester, P.O. Box 88, Manchester, M60 1QD, UK

D. H. Green
School of Electrical & Electronic Engineering, University of Manchester, P.O. Box 88, Manchester, M60 1QD, UK

F. Mavituna
School of Chemical Engineering & Analytical Science, University of Manchester, P.O. Box 88, Manchester, M60 1QD, UK

Abstract

The information needed for the biotic activities of an organism is stored in a coded form in its DNA. This code is universal for all organisms and uses three units called nucleotides, each of which can take one of four possible values to code for twenty different amino acids. Thus it is a mapping from N^3 to P, where N is the set of nucleotides and P is the set of Amino acids. The genetic code has been studied from the points of view of Coding Theory and Information Theory. Here we study it from the point of view of Switching Theory where it is considered as a logic function on a finite field and represented by its Reed-Muller expansion. We first present the genetic code, then develop its Reed-Muller expansion. Potential applications for this approach are also discussed.

1. Introduction

For a cell to grow, divide and carry-out its other activities it needs the information to guide it through these processes. This information is stored in the DNA (Deoxyribonucleic Acid) molecule which can be considered as a blueprint for life.

The DNA molecule famously known as the double helix because of its specially shaped double strands consists of a string of units known as nucleotides. A nucleotide consists of three main chemical components namely, a sugar known as Deoxyribose sugar, a Phosphate group and a base. There are four types of bases known as Adenine, Guanine, Thymine and Cystocine, denoted by A, G, T and C respectively, and leading to four nucleotides with the same symbols. A group of three nucleotides is known as a codon, i.e. it forms a three letter word from a four letter alphabet. Thus there are 64 (= 4^3) possible configurations for a codon, each representing a valid code word. Those words code for Amino acids, which are the building units for proteins. There are twenty amino acids, hence some will have more than one code word leading to robustness against errors.

A series of nucleotides chained together and coding for a functional unit in the cell is known as a gene. Genes normally code for proteins which in turn consist of a string of a large number of Amino acids. For the information stored in the code to be transformed into functions carried out by the proteins, it goes through a certain process.

The information is first copied from the DNA to form another molecule known as the messenger RNA (Ribonucleic Acid) or mRNA; then it is translated by a cell component into the amino acids which are then chained together and processed further to form the active protein.

The purpose of the mRNA is to convey the information from its store (the DNA) to the translation machinery in the cell, akin to a communication system as it effectively transfers a message. The RNA molecule is similar to the DNA except that the sugar is a Ribose sugar and the base Thymine is replaced by another called Uracil denoted by U. This structural difference causes the RNA molecule to be single stranded and much less stable than the DNA as it is not needed after relaying the message. The process of copying the information into the RNA is appropriately known as transcription since it is more or less in the same language, a four letter (nucleotide) alphabet. Converting the code into an amino acid is known as translation since it translates from one language (four nucleotides) to another, the twenty amino acids.

It should be noted that A and G belong to a class of chemicals known as Purines, while C, T and U to another class known as Pyrimidines. The Purines are structurally and functionally closer to each other than to the Pyrimidines, and vice versa. When forming the double stranded DNA molecule, the Purine A pairs with the Pyrimidine T forming what are known as complementary pairs, similarly G pairs with C. [1]

2. The genetic code

The genetic code is the same in all organisms and is thus known as the universal code, Table 1. To transcribe the message, the cell needs to know where to start, in which direction to read and when to stop reading. The first two decisions are determined by structural cues on the DNA molecule related to the gene being transcribed. Stopping is indicated by a stop codon, Table 1. In summary, the genetic code assigns to each base triplet (codon) an amino acid, as such it is a mapping from N^3 to P, where N is the set of nucleotides {A, U, C, G}, and P is the set of Amino acids.

Table 1. The Genetic Code

No.	Base			Amino Acid	
	1st	2nd	3rd	Name	Symbol
1	C	C	C	Proline	**P**
2	C	C	U	Proline	**P**
3	C	C	A	Proline	**P**
4	C	C	G	Proline	**P**
5	C	U	C	Leucine	**L**
6	C	U	U	Leucine	**L**
7	C	U	A	Leucine	**L**
8	C	U	G	Leucine	**L**
9	C	A	C	Histidine	**H**
10	C	A	U	Histidine	**H**
11	C	A	A	Glutamine	**Q**
12	C	A	G	Glutamine	**Q**
13	C	G	C	Argenine	**R**
14	C	G	U	Argenine	**R**
15	C	G	A	Argenine	**R**
16	C	G	G	Argenine	**R**
17	U	C	C	Serine	**S**
18	U	C	U	Serine	**S**
19	U	C	A	Serine	**S**
20	U	C	G	Serine	**S**
21	U	U	C	Phenylalanine	**F**
22	U	U	U	Phenylalanine	**F**
23	U	U	A	Leucine	**L**
24	U	U	G	Leucine	**L**
25	U	A	C	Tyrosine	**Y**
26	U	A	U	Tyrosine	**Y**
27	U	A	A	STOP codon	**Z**
28	U	A	G	STOP codon	**Z**
29	U	G	C	Cysteine	**C**
30	U	G	U	Cysteine	**C**
31	U	G	A	STOP codon	**Z**
32	**U**	**G**	**G**	**Tryptophan**	**W**

The genetic code has been studied from the point of view of Coding Theory [2] and Information Theory [3] where the emphasis is on the representation of the genetic code to investigate its robustness against errors, its information content and how it may have evolved. More recently, it has been viewed as a multiple-valued function on the field of complex numbers [4]. Here we approach the genetic code from the point of view of Switching Theory [5] where it is viewed as a logic function on a finite field. From such a perspective, Table1 can be viewed as a truth table of a multiple-valued logic combinational function which maps three four-valued variables, (i.e. 64 combinations) to a set of 21 values (20 amino acids and the STOP value). We have considered the STOP codon as an output since it has a functional role which is to indicate the end of a gene. We have given it the admittedly non-standard symbol Z. The first column in Table 1 is meant only to keep track of the number of combinations of the three nucleotides, and has no mathematical significance. Also the words base and nucleotide are often used interchangeably within this context.

Table 1 continued

No.	Base			Amino Acid	
	1st	2nd	3rd	Name	Symbol
33	A	C	C	Threonine	**T**
34	A	C	U	Threonine	**T**
35	A	C	A	Threonine	**T**
36	A	C	G	Threonine	**T**
37	A	U	C	Isoleucine	**I**
38	A	U	U	Isoleucine	**I**
39	A	U	A	Isoleucine	**I**
40	A	U	G	Methionine	**M**
41	A	A	C	Asparagine	**N**
42	A	A	U	Asparagine	**N**
43	A	A	A	Lysine	**K**
44	A	A	G	Lysine	**K**
45	A	G	C	Serine	**S**
46	A	G	U	Serine	**S**
47	A	G	A	Argenine	**R**
48	A	G	G	Argenine	**R**
49	G	C	C	Alanine	**A**
50	G	C	U	Alanine	**A**
51	G	C	A	Alanine	**A**
52	G	C	G	Alanine	**A**
53	G	U	C	Valine	**V**
54	G	U	U	Valine	**V**
55	G	U	A	Valine	**V**
56	G	U	G	Valine	**V**
57	G	A	C	Aspartic	**D**
58	G	A	U	Aspartic	**D**
59	G	A	A	Glutamic	**E**
60	G	A	G	Glutamic	**E**
61	G	G	C	Glycine	**G**
62	G	G	U	Glycine	**G**
63	G	G	A	Glycine	**G**
64	**G**	**G**	**G**	**Glycine**	**G**

From a geometrical point of view, each codon can be regarded as a point in a three dimensional space over GF(4) where each point in the space corresponds to an amino acid (or a STOP codon). As discussed above some amino acids have more than one code word (synonyms), i.e. there is redundancy in the code. Thus the amino acids partition the set of codons into equivalent classes.

3. Reed-Muller expansion of the genetic code

For a Galois field GF(q) whose elements are $\{e_0, e_1, \ldots, e_{q-1}\}$, the Reed-Muller (RM) expansion of any function $f(x)$ from GF(q) to GF(q) is a polynomial in x with degree less than q given by [6]

$$f(x) = \sum_{i=0}^{q-1} F(i)x^i$$

where F(i) is given by

$$F(0) = f(e_0)$$

$$F(i) = \sum_{k=1}^{q-1} e_k^{-i}[f(e_0) - f(e_k)]$$

This can be viewed as a Fourier type of transform. It can also be put in the more familiar form [7]

$$f(x) = \sum_{i=0}^{q-1} c_i x^i$$

$$= c_0 + c_1 x + c_2 x^2 + \ldots + c_{q-1} x^{q-1}$$

which is a truncated power series that can be viewed as a Taylor series type of expansion. The coefficients c_i belong to GF(q) and the operations are performed in the GF(q) arithmetic. There are several algorithms for computing the coefficients with different merits such as improving computational efficiency and reducing complexity [8, 9].

For a function of n variables where each variable belongs to GF(q), the function $f(x)$ is a mapping from $[GF(q)]^n$ to GF(q) and the RM expansion will have q^n terms that include product terms of the powers of the different variables [7].

To represent the genetic code by a Reed-Muller expansion, we need to have both the input and the output values belong to the same GF(q). Whilst the inputs belong to GF(4), the smallest finite field that can directly accommodate the outputs is GF(64), which is the smallest power of 4 (the number of nucleotides) to contain the number 21 (the number of amino acids and STOP codon). This means that to be able to formulate an RM expansion for this function we will need to extend GF(4) to GF(64). There are several ways to construct GF(64), namely as GF(2^6), GF(4^3) or GF(8^2). All will give different representations for the elements of the field and consequently different RM expansions, but the same values for the function.

Before constructing the field however, we first need to identify the four nucleotides with the four elements of GF(4) to get a finite field. This will depend on which construction of GF(64) we are going to adopt. We will use GF(2^6) as it simplifies the computations considerably, since on GF(2) and its extensions each element is its own additive inverse. Each of the four nucleotides has to be coded as a binary number, thus two binary digits are needed.

We used an intuitive approach to this coding, simply by arranging the nucleotides by their molecular weight. We chose the most significant binary digit for each nucleotide to code for the chemical class with 0 indicating a Pyrimidine (because they have lower molecular weights) and 1 indicating a Purine. Within a chemical class, we chose the least significant binary digit such that a 0 indicates the nucleotide with the lower molecular weight. Hence the four nucleotides arranged in increasing molecular weight {C, U, A, G} are represented by the increasing binary numbers {00, 01, 10, 11} respectively.

This has the added benefit of preserving the complementary nature of the nucleotides, since the sum of any two complementary pair in this code is 11 (remember from the nature of the genetic code that A and U pair together while C and G pair together). In general, coding of the nucleotides and the resulting coding of the amino acids can be selected to reflect the structural and functional properties of the amino acids, such as chemical groups, bonds, charge, hydrophobic/hydrophilic properties; several coding schemes are possible [10].

We will also need to map the 21 values of the outputs to 21 of the 64 values of GF(64). For simplicity, we chose to assign them in the order they appear in the genetic code, Table1.

To construct GF(64), any of its several primitive polynomials can be used. We have chosen $f(x) = x^6 + x + 1$, and constructed the field from the powers of the corresponding primitive element a [11]. The elements of the field, their binary codes and the amino acids (represented in GF(64)) corresponding to each element are shown in Table 2. Each element in the field is represented by the polynomial

$p(a) = b_5 a^5 + b_4 a^4 + b_3 a^3 + b_2 a^2 + b_1 a^1 + b_0 a^0$

where the b_is belong to {0,1}, with b_0 the least significant digit in the "Binary" column of Table 2. This is re-arranged in ascending powers of the inputs to give the more intuitive Table 3. Using this truth table and the summation transform above, the RM expansion of the genetic code is obtained in Table 4 which shows the powers of x and their corresponding coefficients in GF(64) represented as powers of a.

Table 2. Truth table of the Genetic Code as a function on GF(64)

No.	Input							Output GF(64)
	Binary						GF(64)	
1	0	0	0	0	0	0	**0**	**0**
2	0	0	0	0	0	1	**1**	**0**
3	0	0	0	0	1	0	**a**	**0**
4	0	0	0	0	1	1	$\mathbf{a^6}$	**0**
5	0	0	0	1	0	0	$\mathbf{a^2}$	**1**
6	0	0	0	1	0	1	$\mathbf{a^{12}}$	**1**
7	0	0	0	1	1	0	$\mathbf{a^7}$	**1**
8	0	0	0	1	1	1	$\mathbf{a^{26}}$	**1**
9	0	0	1	0	0	0	$\mathbf{a^3}$	**a**
10	0	0	1	0	0	1	$\mathbf{a^{32}}$	**a**
11	0	0	1	0	1	0	$\mathbf{a^{13}}$	$\mathbf{a^2}$
12	0	0	1	0	1	1	$\mathbf{a^{35}}$	$\mathbf{a^2}$
13	0	0	1	1	0	0	$\mathbf{a^8}$	$\mathbf{a^3}$
14	0	0	1	1	0	1	$\mathbf{a^{48}}$	$\mathbf{a^3}$
15	0	0	1	1	1	0	$\mathbf{a^{27}}$	$\mathbf{a^3}$
16	0	0	1	1	1	1	$\mathbf{a^{18}}$	$\mathbf{a^3}$
17	0	1	0	0	0	0	$\mathbf{a^4}$	$\mathbf{a^4}$
18	0	1	0	0	0	1	$\mathbf{a^{24}}$	$\mathbf{a^4}$
19	0	1	0	0	1	0	$\mathbf{a^{33}}$	$\mathbf{a^4}$
20	0	1	0	0	1	1	$\mathbf{a^{16}}$	$\mathbf{a^4}$
21	0	1	0	1	0	0	$\mathbf{a^{14}}$	$\mathbf{a^5}$
22	0	1	0	1	0	1	$\mathbf{a^{52}}$	$\mathbf{a^5}$
23	0	1	0	1	1	0	$\mathbf{a^{36}}$	**1**
24	0	1	0	1	1	1	$\mathbf{a^{54}}$	**1**
25	0	1	1	0	0	0	$\mathbf{a^9}$	$\mathbf{a^6}$
26	0	1	1	0	0	1	$\mathbf{a^{45}}$	$\mathbf{a^6}$
27	0	1	1	0	1	0	$\mathbf{a^{49}}$	$\mathbf{a^7}$
28	0	1	1	0	1	1	$\mathbf{a^{38}}$	$\mathbf{a^7}$
29	0	1	1	1	0	0	$\mathbf{a^{28}}$	$\mathbf{a^8}$
30	0	1	1	1	0	1	$\mathbf{a^{41}}$	$\mathbf{a^8}$
31	0	1	1	1	1	0	$\mathbf{a^{19}}$	$\mathbf{a^7}$
32	**0**	**1**	**1**	**1**	**1**	**1**	$\mathbf{a^{56}}$	$\mathbf{a^9}$

Table 2 continued

No.	Input							Output GF(64)
	Binary						GF(64	
33	1	0	0	0	0	0	$\mathbf{a^5}$	$\mathbf{a^{10}}$
34	1	0	0	0	0	1	$\mathbf{a^{62}}$	$\mathbf{a^{10}}$
35	1	0	0	0	1	0	$\mathbf{a^{25}}$	$\mathbf{a^{10}}$
36	1	0	0	0	1	1	$\mathbf{a^{11}}$	$\mathbf{a^{10}}$
37	1	0	0	1	0	0	$\mathbf{a^{34}}$	$\mathbf{a^{11}}$
38	1	0	0	1	0	1	$\mathbf{a^{31}}$	$\mathbf{a^{11}}$
39	1	0	0	1	1	0	$\mathbf{a^{17}}$	$\mathbf{a^{11}}$
40	1	0	0	1	1	1	$\mathbf{a^{47}}$	$\mathbf{a^{12}}$
41	1	0	1	0	0	0	$\mathbf{a^{15}}$	$\mathbf{a^{13}}$
42	1	0	1	0	0	1	$\mathbf{a^{23}}$	$\mathbf{a^{13}}$
43	1	0	1	0	1	0	$\mathbf{a^{53}}$	$\mathbf{a^{14}}$
44	1	0	1	0	1	1	$\mathbf{a^{51}}$	$\mathbf{a^{14}}$
45	1	0	1	1	0	0	$\mathbf{a^{37}}$	$\mathbf{a^4}$
46	1	0	1	1	0	1	$\mathbf{a^{44}}$	$\mathbf{a^4}$
47	1	0	1	1	1	0	$\mathbf{a^{55}}$	$\mathbf{a^3}$
48	1	0	1	1	1	1	$\mathbf{a^{40}}$	$\mathbf{a^3}$
49	1	1	0	0	0	0	$\mathbf{a^{10}}$	$\mathbf{a^{15}}$
50	1	1	0	0	0	1	$\mathbf{a^{61}}$	$\mathbf{a^{15}}$
51	1	1	0	0	1	0	$\mathbf{a^{46}}$	$\mathbf{a^{15}}$
52	1	1	0	0	1	1	$\mathbf{a^{30}}$	$\mathbf{a^{15}}$
53	1	1	0	1	0	0	$\mathbf{a^{50}}$	$\mathbf{a^{16}}$
54	1	1	0	1	0	1	$\mathbf{a^{22}}$	$\mathbf{a^{16}}$
55	1	1	0	1	1	0	$\mathbf{a^{39}}$	$\mathbf{a^{16}}$
56	1	1	0	1	1	1	$\mathbf{a^{43}}$	$\mathbf{a^{16}}$
57	1	1	1	0	0	0	$\mathbf{a^{29}}$	$\mathbf{a^{17}}$
58	1	1	1	0	0	1	$\mathbf{a^{60}}$	$\mathbf{a^{17}}$
59	1	1	1	0	1	0	$\mathbf{a^{42}}$	$\mathbf{a^{18}}$
60	1	1	1	0	1	1	$\mathbf{a^{21}}$	$\mathbf{a^{18}}$
61	1	1	1	1	0	0	$\mathbf{a^{20}}$	$\mathbf{a^{19}}$
62	1	1	1	1	0	1	$\mathbf{a^{59}}$	$\mathbf{a^{19}}$
63	1	1	1	1	1	0	$\mathbf{a^{57}}$	$\mathbf{a^{19}}$
64	**1**	**1**	**1**	**1**	**1**	**1**	$\mathbf{a^{58}}$	$\mathbf{a^{19}}$

4. Conclusion

The motivation for considering such a modelling approach to the genetic code is both theoretic out of academic interest, and for potential practical applications [12]. Research in molecular biology (the study of biology at a molecular level such as genes and proteins) is expanding rapidly. This has been greatly catalysed by the development in advanced analytical instrumentation and algorithms, leading to the proliferation of the so called "omics" disciplines, including genomics (study of complete genomes, which involves sequencing of the nucleotides) and proteomics (study of protein structure and function, which is dependant on its amino acid make-up). This is coupled with advancement in algorithms and software, and databases for the analysis of such omics data, embodied in the field of Bioinformatics. Both of those, biological analytical instrumentation and Bioinformatics provide potential application areas for the modelling approach to the genetic code outlined above. In particular, the input to such a model can be a sequence of nucleotides produced by a sequencing instrument. The model would then compute the corresponding sequence of amino acids that can be outputted into a proteomic device or Bioinformatics package giving the final protein structure, hence providing an alternative to the look-up table approach. This can also be part of a more sophisticated information processing system that can compare proteins across organisms to try to infer the function of one from the other, part of the field of functional genomics which is another area of active genomic research. The above model can be implemented in software, or in hardware either embedded in a real time system or on a dedicated microchip [13].

Table 3. The truth table of Table 2 with the input arranged in increasing powers of the primitive element *a*

Input	Output		Input	Output
0	**0**		$\mathbf{a^{31}}$	$\mathbf{a^{11}}$
1	**0**		$\mathbf{a^{32}}$	**a**
a	**0**		$\mathbf{a^{33}}$	$\mathbf{a^{4}}$
$\mathbf{a^{2}}$	**1**		$\mathbf{a^{34}}$	$\mathbf{a^{11}}$
$\mathbf{a^{3}}$	**a**		$\mathbf{a^{35}}$	$\mathbf{a^{2}}$
$\mathbf{a^{4}}$	$\mathbf{a^{4}}$		$\mathbf{a^{36}}$	**1**
$\mathbf{a^{5}}$	$\mathbf{a^{10}}$		$\mathbf{a^{37}}$	$\mathbf{a^{4}}$
$\mathbf{a^{6}}$	**0**		$\mathbf{a^{38}}$	$\mathbf{a^{7}}$
$\mathbf{a^{7}}$	**1**		$\mathbf{a^{39}}$	$\mathbf{a^{16}}$
$\mathbf{a^{8}}$	$\mathbf{a^{3}}$		$\mathbf{a^{40}}$	$\mathbf{a^{3}}$
$\mathbf{a^{9}}$	$\mathbf{a^{6}}$		$\mathbf{a^{41}}$	$\mathbf{a^{8}}$
$\mathbf{a^{10}}$	$\mathbf{a^{15}}$		$\mathbf{a^{42}}$	$\mathbf{a^{18}}$
$\mathbf{a^{11}}$	$\mathbf{a^{10}}$		$\mathbf{a^{43}}$	$\mathbf{a^{16}}$
$\mathbf{a^{12}}$	**1**		$\mathbf{a^{44}}$	$\mathbf{a^{4}}$
$\mathbf{a^{13}}$	$\mathbf{a^{2}}$		$\mathbf{a^{45}}$	$\mathbf{a^{6}}$
$\mathbf{a^{14}}$	$\mathbf{a^{5}}$		$\mathbf{a^{46}}$	$\mathbf{a^{15}}$
$\mathbf{a^{15}}$	$\mathbf{a^{13}}$		$\mathbf{a^{47}}$	$\mathbf{a^{12}}$
$\mathbf{a^{16}}$	$\mathbf{a^{4}}$		$\mathbf{a^{48}}$	$\mathbf{a^{3}}$
$\mathbf{a^{17}}$	$\mathbf{a^{11}}$		$\mathbf{a^{49}}$	$\mathbf{a^{7}}$
$\mathbf{a^{18}}$	$\mathbf{a^{3}}$		$\mathbf{a^{50}}$	$\mathbf{a^{16}}$
$\mathbf{a^{19}}$	$\mathbf{a^{7}}$		$\mathbf{a^{51}}$	$\mathbf{a^{14}}$
$\mathbf{a^{20}}$	$\mathbf{a^{19}}$		$\mathbf{a^{52}}$	$\mathbf{a^{5}}$
$\mathbf{a^{21}}$	$\mathbf{a^{18}}$		$\mathbf{a^{53}}$	$\mathbf{a^{14}}$
$\mathbf{a^{22}}$	$\mathbf{a^{16}}$		$\mathbf{a^{54}}$	**1**
$\mathbf{a^{23}}$	$\mathbf{a^{13}}$		$\mathbf{a^{55}}$	$\mathbf{a^{3}}$
$\mathbf{a^{24}}$	$\mathbf{a^{4}}$		$\mathbf{a^{56}}$	$\mathbf{a^{9}}$
$\mathbf{a^{25}}$	$\mathbf{a^{10}}$		$\mathbf{a^{57}}$	$\mathbf{a^{19}}$
$\mathbf{a^{26}}$	**1**		$\mathbf{a^{58}}$	$\mathbf{a^{19}}$
$\mathbf{a^{27}}$	$\mathbf{a^{3}}$		$\mathbf{a^{59}}$	$\mathbf{a^{19}}$
$\mathbf{a^{28}}$	$\mathbf{a^{8}}$		$\mathbf{a^{60}}$	$\mathbf{a^{17}}$
$\mathbf{a^{29}}$	$\mathbf{a^{17}}$		$\mathbf{a^{61}}$	$\mathbf{a^{15}}$
$\mathbf{a^{30}}$	$\mathbf{a^{15}}$		$\mathbf{a^{62}}$	$\mathbf{a^{10}}$

Table 4. Coefficients of the different powers of x in the RM expansion of the genetic code, as elements in GF(64)

Power of x	$\mathbf{x^0}$	**x**	$\mathbf{x^2}$	$\mathbf{x^3}$	$\mathbf{x^4}$	$\mathbf{x^5}$	$\mathbf{x^6}$	$\mathbf{x^7}$
Coefficient	0	a^{32}	0	a^{2}	a^{38}	a^{33}	a^{60}	a^{62}
Power of x	$\mathbf{x^8}$	$\mathbf{x^9}$	$\mathbf{x^{10}}$	$\mathbf{x^{11}}$	$\mathbf{x^{12}}$	$\mathbf{x^{13}}$	$\mathbf{x^{14}}$	$\mathbf{x^{15}}$
Coefficient	a^{11}	a^{34}	a^{2}	a^{26}	a^{10}	a^{22}	a^{11}	a^{56}
Power of x	$\mathbf{x^{16}}$	$\mathbf{x^{17}}$	$\mathbf{x^{18}}$	$\mathbf{x^{19}}$	$\mathbf{x^{20}}$	$\mathbf{x^{21}}$	$\mathbf{x^{22}}$	$\mathbf{x^{23}}$
Coefficient	a^{50}	a^{43}	a^{43}	a^{42}	a^{6}	a^{11}	a^{4}	a^{36}
Power of x	$\mathbf{x^{24}}$	$\mathbf{x^{25}}$	$\mathbf{x^{26}}$	$\mathbf{x^{27}}$	$\mathbf{x^{28}}$	$\mathbf{x^{29}}$	$\mathbf{x^{30}}$	$\mathbf{x^{31}}$
Coefficient	a^{41}	a^{45}	a^{35}	a^{62}	a^{29}	a^{4}	a^{18}	a^{29}
Power of x	$\mathbf{x^{32}}$	$\mathbf{x^{33}}$	$\mathbf{x^{34}}$	$\mathbf{x^{35}}$	$\mathbf{x^{36}}$	$\mathbf{x^{37}}$	$\mathbf{x^{38}}$	$\mathbf{x^{39}}$
Coefficient	a^{58}	a^{30}	a^{43}	a^{42}	a^{53}	a^{2}	a^{14}	a
Power of x	$\mathbf{x^{40}}$	$\mathbf{x^{41}}$	$\mathbf{x^{42}}$	$\mathbf{x^{43}}$	$\mathbf{x^{44}}$	$\mathbf{x^{45}}$	$\mathbf{x^{46}}$	$\mathbf{x^{47}}$
Coefficient	a^{56}	a^{60}	a^{48}	a^{56}	a^{32}	a^{54}	a^{3}	a^{56}
Power of x	$\mathbf{x^{48}}$	$\mathbf{x^{49}}$	$\mathbf{x^{50}}$	$\mathbf{x^{51}}$	$\mathbf{x^{52}}$	$\mathbf{x^{53}}$	$\mathbf{x^{54}}$	$\mathbf{x^{55}}$
Coefficient	a	a^{7}	a^{14}	a^{36}	a^{9}	a^{49}	0	a^{15}
Power of x	$\mathbf{x^{56}}$	$\mathbf{x^{57}}$	$\mathbf{x^{58}}$	$\mathbf{x^{59}}$	$\mathbf{x^{60}}$	$\mathbf{x^{61}}$	$\mathbf{x^{62}}$	$\mathbf{x^{63}}$
Coefficient	a^{4}	a^{24}	a^{44}	a^{6}	a^{44}	a^{53}	0	a^{29}

Another area of biology that is gaining prominence is synthetic biology which involves designing systems using biological components in particular biomolecules such as DNA and proteins [14]. Yet another area is that of nanobiotechnology, again used for analytical or therapeutic purposes, e.g. for drug delivery or for imaging [15]. A potential application of the model above when coupled with some form of biosensor – another active research area - is as an *in-situ* genomic analysis tool.

In addition to the application, there are theoretically interesting issues relating to the genetic code that fall more in the realm of information theory than of logic design, in particular multi-level codes. Thus another purpose of this paper – further to the RM formulation of the genetic code - is to draw attention to those theoretical and practical areas, notably those that are interdisciplinary in nature.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, D. Bray, K. Hopkin, K. Roberts, and P. Walter, *Essential cell biology*, 2 ed. New York: Garland Science, 2004.

[2] E. E. May, M. A. Vouk, D. L. Bitzer, and D. I. Rosnick, "An error-correcting code framework for genetic sequence analysis," *Journal of the Franklin Institute*, vol. 341, pp. 89-109, 2004.

[3] H. P. Yockey, *Information theory, evolution, and the origin of life*. Cambridge: Cambridge University Press, 2005.

[4] I. Aizenberg and C. Moraga, "The Genetic Code as a Multiple-Valued Function and Its Implementation Using Multilayer Neural Network Based on Multi-Valued Neurons," Proceedings 37th IEEE International Symposium on Multiple-Valued Logic, ISMVL 2007, pp. 2007.

[5] T. Sasao, *Switching theory for logic synthesis* Boston, Mass. ; London: Kluwer Academic Publishers, 1999.

[6] K. S. Menger, "A Transform for Logic Networks," *IEEE Transactions on Computers*, vol. C 18, pp. 241-250, 1969.

[7] D. H. Green and I. S. Taylor, " Modular Representation of Multiple-Valued Logic Systems," *Proceedings of the Institution of Electrical Engineers (London)*, vol. 121, pp. 409-418, 1974.

[8] D. Jankovic, R. S. Stankovic, and R. Drechsler, "Efficient calculation of fixed-polarity polynomial expressions for multiple-valued logic functions," Proceedings 32nd IEEE International Symposium on Multiple-Valued Logic, ISMVL 2002, pp. 76-82, 2002.

[9] B. Harking and C. Moraga, "Efficient derivation of Reed-Muller expansions in multiple-valued logic systems," Proceedings

22nd IEEE International Symposium on Multiple-Valued Logic, ISMVL 1992, pp. 436-441, 1992.

[10] S. Morimoto, "A periodic table for genetic codes," *Journal of Mathematical Chemistry*, vol. 32, pp. 159-200, 2002.

[11] S. Lin and D. J. Costello, *Error control coding : fundamentals and applications*. Englewood Cliffs, N.J.; London Prentice-Hall, 1983.

[12] R. S. Stankovic, C. Moraga, and J. Astola, "Derivatives for multiple-valued functions induced by Galois field and Reed-Muller-Fourier expressions," Proceedings 34th IEEE International Symposium on Multiple-Valued Logic, ISMVL 2004, pp. 184-189, 2004.

[13] V. A. Kalinnikov, "Application of multiple-valued logic in digital technology (review)," *Instruments and Experimental Techniques*, vol. 49, pp. 743-751, 2006.

[14] S. A. Benner and A. M. Sismour, "Synthetic biology," *Nature Reviews Genetics*, vol. 6, pp. 533-543, 2005.

[15] R. S. Kane and A. D. Stroock, "Nanobiotechnology: Protein-nanomaterial interactions," *Biotechnology Progress*, vol. 23, pp. 316-319, 2007.

Appendix II: Published Paper 2

This appendix contains a paper presented at the 38th International Symposium on Multiple-Valued Logic, 2008 (ISMVL '08), held in Houston, Texas in the United States, 22-24 May 2008, and published in its proceedings. The symposium was organised by the Institute of Electrical and Electronic Engineers Inc. (IEEE), (Aleem *et al.* 2008).
(Note that the page numbers have been changed from the original source to match this report)

The paper provides an example of a multiple-valued gene regulatory function where the gene expression levels are multiple-valued as opposed to binary as in the case of phage lambda in chapter seven. Such a situation often occurs in morphogenesis (explained in chapter two).
In the attaché paper we have modelled a regulatory process in the formation of the sense organs in the fruit fly *Drosophila*. This example demonstrates two features of multiple-valued models that were not present in the phage lambda case. Firstly that the output can also be multiple-valued, not just the input, and secondly it presents a case of two multiple-valued inputs.

A Galois Field Approach to Modelling Gene Expression Regulation

H. A. Aleem
School of Chemical Engineering & Analytical Science, University of Manchester, P.O. Box 88, Manchester, M60 1QD, UK

F. Mavituna
School of Chemical Engineering & Analytical Science, University of Manchester, P.O. Box 88, Manchester, M60 1QD, UK

D. H. Green
School of Electrical & Electronic Engineering, University of Manchester, P.O. Box 88, Manchester, M60 1QD, UK

Abstract

Gene expression is the process by which the cell transforms the information in the DNA to functions, often carried out by proteins. Which genes will be expressed, depends on many factors some internal to the cell and others from the environment around it. This can be considered as a logic function prescribing gene expression in response to the different conditions. Apart from continuous models, the most common formulation of such a function is using Boolean logic. This has the major drawback of restricting analysis to binary valued variable which are not always an appropriate approximation. We propose a multiple-valued logic modelling approach on a Galois field and formulate the regulatory function using the Reed-Muller expansion. Two examples are given that illustrate the application of this approach

1. Introduction

The proliferation in the amount and type of information about the processes in a living cell or an organism in general, collectively referred to as "omics", necessitates a systems approach to understanding such processes. Indeed this has led to the development of the discipline of Systems Biology which applies concepts from Systems theory, in particular modelling, to biology in order to comprehend the complex interactions between such processes [1].

Of particular importance are the different stages of gene expression (explained below) and its regulation. Those have been modelled using different mathematical approaches. One popular approach is based on Boolean logic, but it suffers from the shortcoming of only allowing two states for a variable, which is not adequate for modelling some situations. We propose a different approach based on multiple-valued logic, and use it to express the regulatory function in the form of a Reed-Muller expansion.

We first give some background on gene expression, and the current Boolean modelling approach, then introduce our method and apply it to examples from the literature.

2. Gene expression

In the course of its lifetime, a living cell requires the availability of a myriad of substances for both structural (from which it is built) and functional (for it to carry-out its activities) roles. Some of these substances may never be used throughout the lifetime of the cell, for example those needed to defend against an attack by a pathogen that may never occur. Hence, instead of actually synthesising every molecule it may need, it is more efficient to just have the ability to synthesise them, i.e. a form of blueprint. This blueprint is the DNA molecule of the cell, and it contains all the necessary information in a coded form arranged into genes. Each gene normally codes for a protein which can then be used either as part of the cell structure, or to perform a function in the cell usually regulatory (either to switch genes on or off, or to catalyse metabolic reactions) [2]. Regulatory proteins often regulate other genes that in turn produce other proteins that may also affect back the genes that produced the original regulatory protein; they may also regulate their own genes (auto-regulation), leading to a complex network of interactions including positive and negative feedback loops.

The process of transforming the information in the genetic code into its final form (functional or otherwise) is known as gene expression. It starts with transcribing the information from DNA to another form called RNA, which is then translated into proteins that are assembled and processed in the necessary way to fulfil their ultimate purpose. Gene expression is regulated at its different stages; of particular importance is regulation at its inception, i.e. transcription regulation, as it determines which genes

will be expressed. This will depend on different factors, for example in response to internal cell requirements, e.g. when a cell is growing it requires different activities than when it is dividing. Another is in response to external signals such as changes in the cell environment, availability of nutrients or in response to hormones (in higher organisms). In addition, in such organisms, for example in humans, the different cells (e.g. muscle, nerve, skin, etc.) originate from a single cell, the fertilised egg and they all have the same DNA, yet different structures and functions. This specialisation is achieved by deciding which genes to express in each cell among all those available in the DNA.

Thus the genetic code in the DNA of a cell can be thought of as an instruction set, only a subset of which will actually be implemented on need basis. In a sense this is similar to a conditional function call in a program. Transcription regulation selects which functions will be implemented in response to the different signals outlined above.

3. Modelling gene expression regulation

Different approaches are taken towards modelling gene expression [3-5]. Continuous models based on differential equations have the benefit of providing quantitative information, but this comes at the cost of complexity especially when non-linear effects and spatial distribution are considered, as such equations are solved numerically. In addition they inevitably involve assumptions both in terms of mechanisms of interaction and in values of parameters that may not always be justified [6]. Hence, a simpler albeit qualitative modelling approach is desirable, one such approach is that based on Boolean logic.

Boolean models of transcription regulation are popular because of their simplicity and intuitive appeal. They are based on the assumption that variables take binary values [7]. For example a gene is either ON or OFF, and a regulatory protein is either activated (a process that enables it to perform its regulatory action) or deactivated. Similarly for effector molecules, i.e. those chemicals that may activate proteins or affect other processes, their concentration is either above or below a given threshold. This assumption however, is one of the major drawbacks of the binary approach as it does not allow for multiple levels for a variable [5]. For example it is not uncommon that different levels of, say, concentration of a molecule may trigger different processes in the cell or organism. These levels may still be discrete in nature, i.e. representing different activation thresholds. The current approach to overcome this problem is to define for each multiple-level variable, a number of dummy binary variables that is equal to the number of thresholds of the original variable. Each of these binary variables would be zero when the original variable is below the corresponding threshold, and one when it is above it [8]. This leads to an unnecessarily large number of variables, and awkward mathematical formulations especially when there are several variables with several levels.

The second major shortcoming of the Boolean approach is that it assumes that all changes in the variables will take place simultaneously, i.e. a synchronous system [5]. Given that there are possibly thousands of processes taking place in the cell at the same time, many of which are interacting, this assumption becomes unrealistic.

4. A multiple-valued logic approach

We propose here the use of a multiple-valued logic approach to modelling transcription regulation. The intricate interactions between the products of the different genes in a feedback manner, requires modelling the gene regulatory system as a sequential network. There are situations however, where only a combinational network would suffice for modelling transcription regulation. This is especially so in the cases of nutrient availability, a common example is that of the utilisation of sugar sources by the bacterium *E. coli*, whereby when glucose is available a group of genes known as the *lac* operon is switched off, and when glucose is not available but lactose is, the operon is switched on [2]. Such a gene regulation mechanism is known as a *cis*-regulatory function and can be modelled by a combinational logic function.

It is well known that when the number of values q that a variable can take is a prime or a power of a prime, then a finite (Galois) field of order q can be constructed for this variable, denoted by GF(q). In this modelling approach to transcription regulation, the actual value of a variable (say concentration of a chemical entity) is not of essence, but rather the number of values it can take and their order. This is because we are only interested whether a given threshold has been crossed, irrespective of its value.

Any function of a variable x defined on GF(q), can be represented by a polynomial of degree up to q-*1* [9, 10].This is the Reed-Muller (RM) expansion of the function, and is given by [11]

$$f(x) = \sum_{i=0}^{q-1} a_i x^i$$
$$= a_0 + a_1 x + a_2 x^2 + \ldots + a_{q-1} x^{q-1}$$

where the coefficients a_i belong to GF(q) and the operations are performed in the GF(q) arithmetic. There are several methods for computing the coefficients [12, 13]. Here we use a simple matrix based method that directly utilises the truth values of the function. We demonstrate this for the case of GF(3) taken form [14], where the RM-expansion is given by

$$f(x) = a_0 + a_1 x + a_2 x^2$$

substituting for the possible values of x (namely 0, 1 and 2) we get

$$d_0 = f(0) = a_0$$
$$d_1 = f(1) = a_0 + a_1 + a_2$$
$$d_2 = f(2) = a_0 + 2a_1 + a_2$$

This can be represented in matrix form as

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

or in compact notation, simply as

$$d = T \cdot a$$

where now the 3x3 matrix T represents the transformation from the a or polynomial domain to the d or truth value domain. T is invertible, and can be inverted in GF(3) to get

$$a = T^{-1} \cdot d$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix}$$

often expressed as

$$a = S \cdot d$$

Thus to transform from the truth values domain to the polynomial domain we multiply the truth vector d by the transformation matrix S, where d is obtained from the specification of the function. This can be extended to a function of n variables where now the transformation matrix is S_n and can be derived in a recursive way from the one variable case by

$$S_n = \begin{bmatrix} S_{n-1} & 0 & 0 \\ 0 & 2S_{n-1} & S_{n-1} \\ 2S_{n-1} & 2S_{n-1} & 2S_{n-1} \end{bmatrix}$$

where S_0 = [1], and calculations are performed using GF(3) arithmetic.

5. Example from the development of organs

The formation of organs in an organism is part of a process known as morphogenesis which is essentially the creation of structure and form in the organism, and involves the expression of different genes as explained above [15]. One method to achieve such differences in expression is through the concentration gradient of chemicals generically known as morphogens. At different distances from its source, the morphogen will have different concentrations, different thresholds of which will trigger the expression of different genes, helping form the organ. An example of this concept is in the patterns on a butterfly or a zebra. Development in general provides a fitting demonstration of the multiple-valued logic approach.

We illustrate the approach by an example from the development of sense organs in the fruit fly *Drosophila*. Details of how the regulatory process works are given in [16]. We pick one particular example from that source concerning the regulation of a gene "*A*" by two mechanisms, self-activation by its own product denoted by "*a*", and lateral inhibition from a neighbouring cell via a membrane receptor "*B*" that produces a signal denoted by "*b*". Both signals "*a*" and "*b*" have three levels; their different combinations lead to three different expression levels for "*A*". Note that here we are only modelling a combinational function, not a sequential one, hence for the purpose of this example we do not take the time course and hence the feedback effects into account. We have summarised the information in the truth table below, Table 1, paraphrased from [16], where we have renamed the variables *a*, *b* and *A* as x_1, x_2 and y respectively to be consistent with our earlier presentation.

Since all the variables take ternary values, y can be represented as a function of the two variables x_1 and x_2 over GF(3) given by [14]

$$f(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_2 + a_4 x_1 x_2 + a_5 x_1^2 x_2 + a_6 x_2^2 + a_7 x_1 x_2^2 + a_8 x_1^2 x_2^2$$

Table 1. Truth table for the regulation of gene *y* by the signals x_1 and x_2

x_1	x_2	y
0	0	1
0	1	1
0	2	0
1	0	2
1	1	1
1	2	0
2	0	2
2	1	2
2	2	1

Applying the transformation matrix approach outlined above and using the truth table of the regulatory function Table 1, we get

$$y = 1 + x_1^{\,2} + 2x_2 + x_2^{\,2} + x_1 x_2^{\,2} + x_1^{\,2} x_2^{\,2}$$

This equation describes the regulatory function in a compact and meaningful way, as the values of the variables relate directly to the thresholds of the different signals involved unlike the case employing dummy variables. In addition, the coefficients and the powers of the different variables are an indication of their relative contribution to the expression level of the gene. This greatly facilitates what-if analysis of the genetic regulatory function by substituting values for the different variables and directly computing the resulting expression levels of the gene, which is more efficient than the look-up table approach. For example, when a variable is set to zero, this can be used to simulate the situation where the gene producing the corresponding signal is "knocked-out", i.e. deleted. Similarly, if a variable represents the different levels of a given nutrient, then this method greatly facilitates the study of different nutrient scenarios, especially where several nutrients are utilised, as it can be employed to find the optimum nutrient composition based on some optimality criterion. One should be careful to remember that calculations are to be performed in GF(q), but those can be automated in a straightforward manner, especially in the case where q is a prime as this becomes simply modulo q arithmetic.

6. Example from the genetic code

The previous example illustrated the application of the RM-expansion in formulating a conditional function, in the sense that it relates conditions to outputs. However, other biological applications are conceivable, since it essentially represents a mapping between two finite fields. We will briefly outline here another application. As explained above, the genetic information is stored in a coded form in the DNA (Deoxyribonucleic Acid) molecule, famously known as the double helix because of its specially shaped double strands. It is made up of a string of units known as nucleotides, each of which consists of three main components, a sugar known as Deoxyribose, a Phosphate group and a base. There are four types of bases known as Adenine, Guanine, Thymine and Cystocine, indicated by A, G, T and C respectively, and leading to four corresponding types of nucleotides normally denoted with the same symbols. A group of three nucleotides is known as a codon which effectively forms a three letter word from a four letter alphabet. Thus there are 64 (= 4^3) possible configurations for a codon, each representing a valid code word. Those words code for Amino acids, which are the building units for proteins. There are twenty amino acids, hence some will have more than one code leading to robustness against errors. A series of nucleotides coding a functional unit in the cell is known as a gene. Genes usually code for proteins which in turn consists of a string of a large number of the twenty possible Amino acids.

The information is first copied from the DNA to form another molecule known as the messenger RNA (Ribonucleic Acid) or mRNA, the purpose of which is to convey the information from its store (the DNA) to the translation machinery in the cell. The RNA molecule is similar to the DNA except that the sugar is a Ribose and the base Thymine is replaced by another called Uracil indicated by U. This structural difference causes the RNA molecule to be single stranded and much less stable than the DNA as it is not needed after relaying the message, and its components will be needed to synthesise other messengers.

The process of copying the information into the RNA is appropriately known as transcription since it is more or less in the same language, a four letter (nucleotide) alphabet. Converting the code into an amino acid is known as translation since it translates from one language (four nucleotides) to another, the twenty amino acids. The genetic code is the same in all organisms and is consequently known as the universal code, presented in Table 2 where each amino acid has a standard symbol [2]. It also contains STOP codons which indicate the end of a gene. The start of a gene is identified by structural cues in the DNA molecule.

Table 2 effectively represents a truth table for a multiple-valued logic combinational function where the inputs are defined on GF(4) and the output on GF(64), thus we can formulate a Reed-Muller expansion for this function. Since the output can have

any of twenty one values (the twenty amino acid and the stop codon), the smallest finite field that can directly accommodate both the output and the inputs is GF(64). Which means that the inputs have to be represented as a variable on GF(64). There are several scenarios for doing that e.g. GF(4^3), GF(8^2) and GF(2^6), all of which will lead to a polynomial of degree up to 63, albeit with different coefficients, from the appropriate base field, and computed using the relevant 64x64 transformation matrix. This is primarily a mechanistic task, and hence will not be pursued here.

This modelling approach can aid in theoretical studies of the genetic code by considering it as a multi-level (non-binary) code where issues of distance between code words, e.g. a Lee type metric [17], and other information theoretic issues can be explored. Furthermore there are several potential practical applications; for example in the areas of Bioinformatics and biological instrumentation, where it can lead to novel genome analysis approaches. Also in the emerging fields of biosensors and bio-nanotechnology where it can lead to *in-situ* genomic analysis tools.

Table 2a. The Genetic Code

No.	Base			Amino Acid	
	1st	2nd	3rd	Name	Symbol
1	**U**	**U**	**U**	Phenylalanine	F
2	**U**	**U**	**C**	Phenylalanine	F
3	**U**	**U**	**A**	Leucine	L
4	**U**	**U**	**G**	Leucine	L
5	**U**	**C**	**U**	Serine	S
6	**U**	**C**	**C**	Serine	S
7	**U**	**C**	**A**	Serine	S
8	**U**	**C**	**G**	Serine	S
9	**U**	**A**	**U**	Tyrosine	Y
10	**U**	**A**	**C**	Tyrosine	Y
11	**U**	**A**	**A**	STOP codon	-
12	**U**	**A**	**G**	STOP codon	-
13	**U**	**G**	**U**	Cysteine	C
14	**U**	**G**	**C**	Cysteine	C
15	**U**	**G**	**A**	STOP codon	-
16	**U**	**G**	**G**	Tryptophan	W
17	**C**	**U**	**U**	Leucine	L
18	**C**	**U**	**C**	Leucine	L
19	**C**	**U**	**A**	Leucine	L
20	**C**	**U**	**G**	Leucine	L
21	**C**	**C**	**U**	Proline	P
22	**C**	**C**	**C**	Proline	P
23	**C**	**C**	**A**	Proline	P
24	**C**	**C**	**G**	Proline	P
25	**C**	**A**	**U**	Histidine	H
26	**C**	**A**	**C**	Histidine	H
27	**C**	**A**	**A**	Glutamine	Q
28	**C**	**A**	**G**	Glutamine	Q
29	**C**	**G**	**U**	Arginine	R
30	**C**	**G**	**C**	Arginine	R
31	**C**	**G**	**A**	Arginine	R
32	**C**	**G**	**G**	Arginine	R

Table 2b. The Genetic Code (cont'd.)

No.	Base			Amino Acid	
	1st	2nd	3rd	Name	Symbol
33	**A**	**U**	**U**	Isoleucine	I
34	**A**	**U**	**C**	Isoleucine	I
35	**A**	**U**	**A**	Isoleucine	I
36	**A**	**U**	**G**	Methionine	M
37	**A**	**C**	**U**	Threonine	T
38	**A**	**C**	**C**	Threonine	T
39	**A**	**C**	**A**	Threonine	T
40	**A**	**C**	**G**	Threonine	T
41	**A**	**A**	**U**	Asparagine	N
42	**A**	**A**	**C**	Asparagine	N
43	**A**	**A**	**A**	Lysine	K
44	**A**	**A**	**G**	Lysine	K
45	**A**	**G**	**U**	Serine	S
46	**A**	**G**	**C**	Serine	S
47	**A**	**G**	**A**	Arginine	R
48	**A**	**G**	**G**	Arginine	R
49	**G**	**U**	**U**	Valine	V
50	**G**	**U**	**C**	Valine	V
51	**G**	**U**	**A**	Valine	V
52	**G**	**U**	**G**	Valine	V
53	**G**	**C**	**U**	Alanine	A
54	**G**	**C**	**C**	Alanine	A
55	**G**	**C**	**A**	Alanine	A
56	**G**	**C**	**G**	Alanine	A
57	**G**	**A**	**U**	Aspartic	D
58	**G**	**A**	**C**	Aspartic	D
59	**G**	**A**	**A**	Glutamic	E
60	**G**	**A**	**G**	Glutamic	E
61	**G**	**G**	**U**	Glycine	G
62	**G**	**G**	**C**	Glycine	G
63	**G**	**G**	**A**	Glycine	G
64	**G**	**G**	**G**	Glycine	G

7. Conclusion

We have presented a novel approach to modelling gene transcription regulation by addressing the more realistic multiple-valued case (as opposed to the binary one) and expressing the regulatory function as a Reed-Muller expansion on a corresponding Galois field. This approach has several advantages including

- Efficiently represents multi-input multiple-valued gene regulatory functions in a compact form. Under the appropriate conditions, it can easily be extended to multi-output functions as well.
- Greatly facilitates what-if analysis as outlined above.
- Calculations can be automated as there are efficient algorithms for deriving the RM-coefficients.
- Because it is employed in logic design, the same concepts can be applied in designing control schemes to control the expression levels of different genes, potentially formalising the process of drug design in a mathematical sense.

In addition to gene regulation expression, we have also outlined the application of the same approach to model the genetic code as a function on a Galois field.

We have restricted the development here to combinational logic functions. Work on sequential logic which addresses the dynamics of gene expression is in progress.

Reference

[1] O. Wolkenhauer, H. Kitano, and K. H. Cho, "Systems biology," *IEEE Control Systems Magazine*, vol. 23, pp. 38-48, 2003.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, D. Bray, K. Hopkin, K. Roberts, and P. Walter, *Essential cell biology*, 2 ed. New York: Garland Science, 2004.

[3] H. de Jong, C. Chaouiya, and D. Thieffry, "Dynamical modeling of biological regulatory networks," *Biosystems*, vol. 84, pp. 77-80, 2006.

[4] L. F. A. Wessels, E. P. Van Someren, and M. J. T. Reinders, "A comparison of genetic network models," presented at Pacific Symposium on Biocomputing, 6:508 - 519, 2001.

[5] P. Smolen, D. A. Baxter, and J. H. Byrne, "Modeling transcriptional control in gene networks - Methods, recent results, and future directions," *Bulletin of Mathematical Biology*, vol. 62, pp. 247-292, 2000.

[6] J. M. G. Vilar, C. C. Guet, and S. Leibler, "Modeling network dynamics: the lac operon, a case study," *Journal of Cell Biology*, vol. 161, pp. 471-476, 2003.

[7] R. Thomas, "Regulatory Networks Seen as Asynchronous Automata - a Logical Description," *Journal of Theoretical Biology*, vol. 153, pp. 1-23, 1991.

[8] E. Snoussi and R. Thomas, "Logical Identification of All Steady-States - the Concept of Feedback Loop Characteristic States," *Bulletin of Mathematical Biology*, vol. 55, pp. 973-991, 1993.

[9] K. S. Menger, "A Transform for Logic Networks," *IEEE Transactions on Computers*, vol. C 18, pp. 241-250, 1969.

[10] D. K. Pradhan, "Theory of Galois Switching Functions," *IEEE Transactions on Computers*, vol. 27, pp. 239-248, 1978.

[11] D. H. Green and I. S. Taylor, "Modular Representation of Multiple-Valued Logic Systems," *Proceedings of the Institution of Electrical Engineers (London)*, vol. 121, pp. 409-418, 1974.

[12] D. Jankovic, R. S. Stankovic, and R. Drechsler, "Efficient calculation of fixed-polarity polynomial expressions for multiple-valued logic functions," ISMVL 2002. Proceedings 32nd IEEE International Symposium on Multiple-Valued Logic , 2002.

[13] B. Harking and C. Moraga, "Efficient derivation of Reed-Muller expansions in multiple-valued logic systems," ISMVL 1992. Proceedings 22nd IEEE International Symposium on Multiple-Valued Logic , 1992.

[14] D. H. Green, "Ternary Reed-Muller Switching-Functions with Fixed and Mixed Polarities," *International Journal of Electronics*, vol. 67, pp. 761-775, 1989.

[15] S. F. Gilbert, *Developmental biology*, 6 ed. Sunderland, Mass.: Sinauer Associates, 2000.

[16] A. Ghysen and R. Thomas, "The formation of sense organs in Drosophila: a logical approach," *Bioessays*, vol. 25, pp. 802-807, 2003.

[17] E. R. Berlekamp, *Algebraic coding theory*. New York ; London (etc.) McGraw-Hill, 1968.

References

Ackers, G. K., Johnson, A. D. & Shea, M. A. (1982). Qunatitative Model for Gene Regluation by Lambda Phabe Repressor. *Proceedings of the National Academy of Sciences of the United States of America-Biological Sciences,* 79(4)**,** 1129-1133.

Akers, S. B. (1959). On a Theory of Boolean Functions. *Journal of the Society for Industrial and Applied Mathematics,* 7(4)**,** 487-498.

Akers, S. B. (1987). The Use of Linear Sums in Exhaustive Testing. *Computers & Mathematics with Applications,* 13(5-6)**,** 475-483.

Akutsu, T., Miyano, S. & Kuhara, S. (1999). Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Pac Symp Biocomput***,** 17-28.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Bray, D., Hopkin, K., Roberts, K. & Walter, P. (2004). *Essential cell biology* (2 ed.). New York: Garland Science.

Aleem, H. A., Green, D. H. & Mavituna, F. (2009). Representing the Genetic Code as a Function on a Galois Field Using the Reed-Muller Expansion. *In:* Proceedings 39th International Symposium on Multiple-Valued Logic, 2009. ISMVL '09. , 2009. 356-361.

Aleem, H. A., Mavituna, F. & Green, D. H. (2008). A Galois Field Approach to Modelling Gene Expression Regulation. *In:* Proceedings 38th International Symposium on Multiple Valued Logic, 2008. ISMVL 2008. , 2008. 88-93.

Almaini, A. E. A. (1994). *Electronic Logic Systems* (3rd ed.): Prentice Hall.

Alon, U. (2007a). *An introduction to systems biology: design principles of biological circuit*. Boca Raton, Fla: Chapman & Hall/CRC.

Alon, U. (2007b). Network motifs: theory and experimental approaches. *Nature Reviews Genetics,* 8(6)**,** 450-461.

Amos, M. (2005). *Theoretical and Experimental DNA Computation*. Berlin Heidelberg: Springer.

Andrianantoandro, E., Basu, S., Karig, D. K. & Weiss, R. (2006). Synthetic biology: new engineering rules for an emerging discipline. *Molecular Systems Biology*, 2.

Arnold, S., Siemann-Herzberg, M., Schmid, J. & Reuss, M. (2005). Model-based inference of gene expression dynamics from sequence information. *Biotechnology for the Future,* 100**,** 89-179.

Atkinson, B. & Mavituna, F. (1991). *Biochemical engineering and biotechnology handbook* (2 ed.). Basingstoke: Macmillan.

Bailey, J. E. (1991). Toward a Science of Metabolic Engineering. *Science,* 252(5013)**,** 1668-1675.

Beauchamp, K. G. (1975). *Walsh functions and their applications*. London: Academic Press.

Beauchamp, K. G. (1987). *Transforms for engineers : a guide to signal processing*. Oxford: Clarendon.

Benjauthrit, B. & Reed, I. S. (1976). Galois Switching Functions and Their Applications. *IEEE Transactions on Computers,* C-25(1)**,** 78-86.

Benjauthrit, B. & Reed, I. S. (1978). Fundamental Structure of Galois Switching Functions. *IEEE Transactions on Computers,* 27(8)**,** 757-762.

Benner, S. A. & Sismour, A. M. (2005). Synthetic biology. *Nature Reviews Genetics,* 6(7)**,** 533-543.

Berlekamp, E. R. (1968). *Algebraic coding theory*. New York ; London (etc.) McGraw-Hill.

Bintu, L., Buchler, N. E., Garcia, H. G., Gerland, U., Hwa, T., Kondev, J. & Phillips, R. (2005). Transcriptional regulation by the numbers: models. *Current Opinion in Genetics & Development,* 15(2)**,** 116-124.

BioIndustry Association (2004). Bioscience 2015. UK Department of Trade and Industry and Department of Health, available at www.bioindustry.org/bigtreport.

Birkhoff, G. & Bartee, T. C. (1970). *Modern applied algebra*. New York: McGraw-Hill.

Brent, R. (2004). A partnership between biology and engineering. *Nature Biotechnology,* 22(10)**,** 1211-1214.

Buchler, N. E., Gerland, U. & Hwa, T. (2003). On schemes of combinatorial transcription logic. *Proceedings of the National Academy of Sciences of the United States of America,* 100(9)**,** 5136-5141.

Camacho, D., Licona, P. V., Mendes, P. & Laubenbacher, R. (2007). Comparison of reverse-engineering methods using an in Silico network. *Reverse Engineering Biological Networks,* 1115**,** 73-89.

Carré, B. (1979). *Graphs and networks*. Oxford: Clarendon Press.

Cassandras, C. G. (1993). *Discrete Event Systems: Modelling and Performance Analysis*: Irwin Inc.

Casti, J. L. (1989). *Alternate realities : mathematical models of nature and man* New York ; Chichester Wiley.

Casti, J. L. & Karlqvist, A. (eds.) (1990). *Beyond belief: randomness, prediction, and explanation in science*, Boca Raton, Fla.: CRC Press.

Caulfield, T., Einsiedel, E., Merz, J. F. & Nicol, D. (2006). Trust, patents and public perceptions: the governance of controversial biotechnology research. *Nature Biotechnology,* 24**,** 1352-1354.

Cha, P. D., Rosenberg, J. J. & Dym, C. L. (2000). *Fundamentals of Modeling and Analyzing Engineering Systems*: Cambridge University Press.

Chaouiya, C., Remy, E. & Thieffry, D. (2008). Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms,* 6(2)**,** 165-177.

Cho, K. H., Choo, S. M., Jung, S. H., Kim, J. R., Choi, H. S. & Kim, J. (2007). Reverse engineering of gene regulatory networks. *IET Systems Biology,* 1(3)**,** 149-163.

Christensen, C., Gupta, A., Maranas, C. D. & Albert, R. (2007). Large-scale inference and graph-theoretical analysis of gene-regulatory networks in B-Subtilis. *Physica A-Statistical Mechanics and Its Applications,* 373**,** 796-810.

Comet, J. P., Klaudel, H. & Liauzu, S. (2005). Modeling multi-valued genetic regulatory networks using high-level Petri Nets. *Applications and Theory of Petri Nets 2005, Proceedings.* Berlin: Springer-Verlag Berlin.

Cornish-Bowden, A. (2005). Making systems biology work in the 21st century. *Genome Biol,* 6(4)**,** 317.

Cornish-Bowden, A. & Cardenas, M. L. (2005). Systems biology may work when we learn to understand the parts in terms of the whole. *Biochemical Society Transactions,* 33**,** 516-519.

Cover, T. M. & Thomas, J. A. (1991). *Elements of information theory*. New York: Wiley.

Cox, D., Little, J. & O'Shea, D. (2007). *Ideals, Varieties, and Algorithms* (3rd ed.): Springer

D'Haeseleer, P., Liang, S. D. & Somogyi, R. (2000). Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics,* 16(8)**,** 707-726.

Damarla, T. R. & Karpovsky, M. (1989). Fault detection in combinational networks by Reed-Muller transforms. *IEEE Transactions on Computers,* 38(6)**,** 788-797.

Davidson, E. H. (2006). *The regulatory genome : gene regulatory networks in development and evolution*. Burlington, Mass: Academic Press/Elsevier.

Davidson, E. H., Rast, J. P., Oliveri, P., Ransick, A., Calestani, C., Yuh, C.-H., Minokawa, T., Amore, G., Hinman, V., Arenas-Mena, C., Otim, O., Brown, C. T., Livi, C. B., Lee, P. Y., Revilla, R., Rust, A. G., Pan, Z. J., Schilstra, M. J., Clarke, P. J. C., Arnone, M. I., Rowen, L., Cameron, R. A., McClay, D. R., Hood, L. & Bolouri, H. (2002). A genomic regulatory network for development. *Science,* 295(5560)**,** 1669-1678.

Davio, M., Deschamps, J. & Thayse, A. (1978). *Discrete and Switching Functions*: McGraw-Hill.

De Jong, H. (2002). Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology,* 9(1)**,** 67-103.

de Lorenzo, V. & Danchin, A. (2008). Synthetic biology: discovering new worlds and new words - The new and not so new aspects of this emerging research field. *Embo Reports,* 9(9)**,** 822-827.

Debnath, D. & Sasao, T. (2000). Exact minimization of fixed polarity Reed-Muller expressions for incompletely specified functions. *In:* Proceedings of the ASP-DAC 2000. Asia and South Pacific Design Automation Conference, 2000. , 2000. 247-252.

Doebelin, E. O. (1980). *System Modeling And Response: Theoretical and Experimental Approaches*: John Wiley & Sons.

Drubin, D. A., Way, J. C. & Silver, P. A. (2007). Designing biological systems. *Genes & Development,* 21(3)**,** 242-254.

Dueber, J. E., Yeh, B. J., Bhattacharyya, R. P. & Lim, W. A. (2004). Rewiring cell signaling: the logic and plasticity of eukaryotic protein circuitry. *Current Opinion in Structural Biology,* 14(6)**,** 690-699.

Duncan, M. W. (2007). Omics and its 15 minutes. *Experimental Biology and Medicine,* 232(4)**,** 471-472.

Endy, D. (2005). Foundations for engineering biology. *Nature,* 438(7067)**,** 449-453.

Erdi, P. & Toth, J. (1989). *Mathematical models of the chemical reaction*: Manchester University Press.

European Biopharmaceutical Enterprises (2007). EBE Annual Highlights 2007/ 2008. European Biopharmaceutical Enterprises available at www.ebe-biopharma.org.

Falkowski, B. J. (1999). A note on the polynomial form of Boolean functions and related topics. *IEEE Transactions on Computers,* 48(8)**,** 860-864.

Falkowski, B. J. & Lozano, C. C. (2005). Quaternary Fixed-Polarity Reed-Muller expansion computation through operations on disjoint cubes and its comparison with other methods. *Computers & Electrical Engineering,* 31(2)**,** 112-131.

Falkowski, B. J., Lozano, C. C. & Rahardja, S. (2005). Recursive algorithm for generation of fixed polarity reed-muller expansions over GF(5). *In:* The 2005 48th Midwest Symposium on Circuits and Systems, 2005. MWSCAS '05., 2005. 191-194.

Falkowski, B. J. & Rahardja, S. (1997). Classification and properties of fast linearly independent logic transformations. *IEEE Transactions on Circuits and Systems II-Analog and Digital Signal Processing,* 44(8)**,** 646-655.

Falkowski, B. J. & Yan, S. (2004). Walsh-Hadamard spectral minimization of fixed polarity Reed-Muller expansions. *In:* The 2004 47th Midwest Symposium on Circuits and Systems, 2004. MWSCAS '04. , 2004. I-509-12 vol.1.

Farley, M. S. & Rouse, W. B. (2000). Technology Challenges \& Opportunities in the Biotechnology, Pharmaceutical \& Medical Device Industries. *Inf. Knowl. Syst. Manag.,* 2(2)**,** 133-141.

Faure, A., Naldi, A., Chaouiya, C. & Thieffry, D. (2006). Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics,* 22(14)**,** E124-E131.

Fell, D. (1997). *Understanding the Control of Metabolism*: Portland Press.

Fowkes, N. D. & Mahony, J. J. (1994). *An Introduction to Mathematical Modelling*: John Wiley & Sons.

Friedman, N., Linial, M., Nachman, I. & Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Journal of Computational Biology,* 7(3-4)**,** 601-620.

Gallian, J. A. (1994). *Contemporary abstract algebra* (3 ed.). Lexington, Mass: D.C. Heath.

Gatherer, D. (2010). So what do we really mean when we say that systems biology is holistic? *BMC Systems Biology,* 4.

Gershenfeld, N. A. (1999). *The nature of mathematical modeling* Cambridge: Cambridge University Press.

Ghysen, A. & Thomas, R. (2003). The formation of sense organs in Drosophila: a logical approach. *Bioessays,* 25(8)**,** 802-807.

Gibson, D. G., Glass, J. I., Lartigue, C., Noskov, V. N., Chuang, R. Y., Algire, M. A., Benders, G. A., Montague, M. G., Ma, L., Moodie, M. M., Merryman, C., Vashee, S., Krishnakumar, R., Assad-Garcia, N., Andrews-Pfannkoch, C., Denisova, E. A., Young, L., Qi, Z. Q., Segall-Shapiro, T. H., Calvey, C. H., Parmar, P. P., Hutchison, C. A., Smith, H. O. & Venter, J. C. (2010). Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome. *Science,* 329(5987)**,** 52-56.

Gil, C. & Ortega, J. (1998). Algebraic test-pattern generation based on the Reed-Muller spectrum. *IEE Proceedings: Computers and Digital Techniques,* 145(4)**,** 308-316.

Gilbert, S. F. (2000). *Developmental biology* (6 ed.). Sunderland, Mass.: Sinauer Associates.

Glass, L. & Kauffman, S. A. (1973). Logical Analysis of Continuous, Nonlinear Biochemical Control Networks. *Journal of Theoretical Biology,* 39(1)**,** 103-129.

Green, D. (1986). *Modern logic design*. Wokingham: Addison-Wesley.

Green, D. H. (1989). Ternary Reed-Muller switching functions with fixed and mixed polarities. *International Journal of Electronics,* 67(5)**,** 761-775.

Green, D. H. (1990a). Reed-Muller canonical forms with mixed polarity and their manipulations. *IEE Proceedings, Part E: Computers and Digital Techniques,* 137(1)**,** 103-113.

Green, D. H. (1990b). Reed-Muller expansions with fixed and mixed polarities over GF(4). *IEE Proceedings, Part E: Computers and Digital Techniques,* 137(5)**,** 380-388.

Green, D. H. & Edkins, M. (1978). Synthesis Procedures for Switching Circuits Represented in Generalised Reed-Muller form over a Finite Field. *IEE Journal on Computers and Digital Techniques,* 1(1)**,** 27-35.

Green, D. H. & Khuwaja, G. A. (1992). Simplification of switching functions expressed in Reed-Muller algebraic form. *IEE Proceedings, Part E: Computers and Digital Techniques,* 139(6)**,** 511-518.

Green, D. H. & Taylor, I. S. (1974). Modular Rerpesentation of Multiple-Valued Logic Systems. *Proceedings of the Institution of Electrical Engineers (London),* 121(6)**,** 409-418.

Guet, C. C., Elowitz, M. B., Hsing, W. H. & Leibler, S. (2002). Combinatorial synthesis of genetic networks. *Science,* 296(5572)**,** 1466-1470.

Habib, M. K. (1993). Efficient and Fast Algorithm to Generate Minimal Reed Muller Exclusive-or Expansions with Mixed Polarity for Completely and Incompletely Specified Functions and Its Computer Implementation. *Computers & Electrical Engineering,* 19(3)**,** 193-211.

Hartwell, L. H., Hopfield, J. J., Leibler, S. & Murray, A. W. (1999). From molecular to modular cell biology. *Nature,* 402(6761)**,** C47-C52.

Hasty, J., McMillen, D. & Collins, J. J. (2002). Engineered gene circuits. *Nature,* 420(6912)**,** 224-230.

Hecker, M., Lambeck, S., Toepfer, S., van Someren, E. & Guthke, R. (2009). Gene regulatory network inference: Data integration in dynamic models-A review. *Biosystems,* 96(1)**,** 86-103.

Heinemann, M. & Panke, S. (2006). Synthetic biology - putting engineering into biology. *Bioinformatics,* 22(22)**,** 2790-2799.

Herrera, S. (2004). Industrial biotechnology - a chance at redemption. *Nature Biotechnology,* 22(6)**,** 671-675.

Hurst, S. L. (1978). *The Logical Processing of Digital Signals*: Crane, Russak & Company Inc., .

Hurst, S. L., Miller, D. M. & Muzio, J. C. (1985). *Spectral Techniques in Digital Logic*: Academic Press.

Hwan Mook, C., Su Young, P. & Rey, S. (1998). The MacLaurin's and Taylor's series expansions of the symbolic multiple valued logic functions. *In:* Proceedings 28th IEEE International Symposium on Multiple-Valued Logic, 1998. ISMVL 1998 1998. 65-70.

Ideker, T. & Lauffenburger, D. (2003). Building with a scaffold: emerging strategies for high- to low-level cellular modeling. *Trends in Biotechnology,* 21(6)**,** 255-262.

Institute for Manufacturing (2007). bioProcessUK roadmap. University of Cambridge available at www.bioprocessuk-website.org/documents.

Istrail, S. & Davidson, E. H. (2005). Logic functions of the genomic cis-regulatory code. *Proceedings of the National Academy of Sciences of the United States of America,* 102(14)**,** 4954-4959.

Jankovic, D., Stankovic, R. S. & Drechsler, R. (2000). Efficient calculation of fixed-polarity polynomial expressions for multiple-valued logic functions. *In:* Proceedings 32nd IEEE International Symposium on Multiple-Valued Logic, 2002. ISMVL 2002. , 2002. 76-82.

Jarrah, A. S., Laubenbacher, R., Stigler, B. & Stillman, M. (2007). Reverse-engineering of polynomial dynamical systems. *Advances in Applied Mathematics,* 39**,** 477-489.

Jha, N. & Gupta, S. (2003). *Testing of Digital Systems*: Cambridge University Press.

Kaplan, S., Bren, A., Zaslaver, A., Dekel, E. & Alon, U. (2008). Diverse two-dimensional input functions control bacterial sugar genes. *Molecular Cell,* 29(6)**,** 786-792.

Kauffman, S., Peterson, C., Samuelsson, B. & Troein, C. (2004). Genetic networks with canalyzing Boolean rules are always stable. *Proceedings of the National Academy of Sciences of the United States of America,* 101(49)**,** 17102-17107.

Kauffman, S. A. (1993). *The Origins of Order: Self-organization and Selection in Evolution*. New York: Oxford University Press.

Kaul, P., Banerjee, A. & U.C.Banerjee. (2004). Opportunities for the pharmaceutical industry: key biotransformation technologies for the future. *In:* Drug Discovery World spring conference, 2004. 80-86.

Kazakos, D. & Papantoni-Kazakos, P. (1990). *Detection and Estimation,* . New York Computer Science Press.

Kell, D. B. & Oliver, S. G. (2004). Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. *Bioessays,* 26(1)**,** 99-105.

Kitano, H. (2002). Computational systems biology. *Nature,* 420(6912)**,** 206-210.

Kraniauskas, P. (1992). *Transforms in Signals and Systems*: Pearson.

Kremling, A., Kremling, S. & Bettenbrock, K. (2009). Catabolite repression in Escherichia coli- a comparison of modelling approaches. *FEBS Journal,* 276(2)**,** 594-602.

Lahdesmaki, H., Shmulevich, I. & Yli-Harja, O. (2003). On learning gene regulatory networks under the Boolean network model. *Machine Learning,* 52(1-2)**,** 147-167.

Lahteenmaki, R. & Lawrence, S. (2006). Public biotechnology 2005 - the numbers. *Nature Biotechnology,* 24(6)**,** 625-634.

Lala, P. K. (1997). *Digital circuit testing and testability*. London Academic Press.

Laubenbacher, R. & Stigler, B. (2004). A computational algebra approach to the reverse engineering of gene regulatory networks. *Journal of Theoretical Biology,* 229(4)**,** 523-537.

Lay, J. O., Borgmann, S., Liyanage, R. & Wilkins, C. L. (2006). Problems with the "omics". *Trac-Trends in Analytical Chemistry,* 25(11)**,** 1046-1056.

Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., Hannett, N. M., Harbison, C. T., Thompson, C. M., Simon, I., Zeitlinger, J., Jennings, E. G., Murray, H. L., Gordon, D. B., Ren, B., Wyrick, J. J., Tagne, J. B., Volkert, T. L., Fraenkel, E., Gifford, D. K. & Young, R. A. (2002). Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science,* 298(5594)**,** 799-804.

Lesne, A. (2006). Complex networks: From graph theory to biology. *Letters in Mathematical Physics,* 78(3)**,** 235-262.

Lidl, R. (1994). *Introduction to finite fields and their applications*: Cambridge University Press.

Lin, S. & Costello, D. J. (1983). *Error control coding : fundamentals and applications*. Englewood Cliffs, N.J.; London Prentice-Hall.

Ljung, L. (1998). *System Identification: Theory for the User* Prentice Hall.

Lorenz, P. & Eck, J. (2005). Metagenomics and industrial applications. *Nature Reviews Microbiology,* 3(6)**,** 510-516.

Lorkowski, S. & Cullen, P. (eds.) (2003). *Analysing Gene Expression: A Handbook of Methods: Possibilities and Pitfalls* Wiley-VCH Verlag GmbH & Co.

Marchisio, M. A. & Stelling, J. (2008). Computational design of synthetic gene circuits with composable parts. *Bioinformatics,* 24(17)**,** 1903-1910.

Martinez-Antonio, A., Janga, S. C. & Thieffry, D. (2008). Functional organisation of Escherichia coli transcriptional regulatory network. *Journal of Molecular Biology,* 381(1)**,** 238-247.

May, E. E., Vouk, M. A., Bitzer, D. L. & Rosnick, D. I. (2004). An error-correcting code framework for genetic sequence analysis. *Journal of the Franklin Institute,* 341(1-2)**,** 89-109.

Mayo, A. E., Setty, Y., Shavit, S., Zaslaver, A. & Alon, U. (2006). Plasticity of the cis-regulatory input function of a gene. *PLoS Biology,* 4(4)**,** 555-561.

McCluskey, E. J. (1986). *Logic Design Principles*: Prentice Hall.

McEliece, R. J. (1987). *Finite fields for computer scientists and engineers* (Vol. 23). Boston, Mass. ; Lancaster Kluwer.

McKenzie, L., Almaini, A. E. A., Miller, J. F. & Thomson, P. (1993). Optimization of Reed-Muller Logic Functions. *International Journal of Electronics,* 75(3)**,** 451-466.

Mehta, T. S., Zakharkin, S. O., Gadbury, G. L. & Allison, D. B. (2006). Epistemological issues in omics and high-dimensional biology: give the people what they want. *Physiological Genomics,* 28(1)**,** 24-32.

Menger, K. S. (1969). A Transform for Logic Networks. *IEEE Transactions on Computers,* C 18(3)**,** 241-250.

Muller, D. E. (1954). Application of Boolean algebra to switching circuit design and to error detection. *Institute of Radio Engineers -- Transactions of Professional Group on Electronic Computers,* EC-3(3)**,** 6-12.

Murata, T. (1989). Petrinets - Properties, Analaysis and Applications. *Proceedings of the IEEE,* 77(4)**,** 541-580.

Nagel, E. (1961). *The structure of science: problems in the logic of scientific explanation*. London: Routledge and Kegan Paul.

Narang, A. (2006). Comparative analysis of some models of gene regulation in mixed-substrate microbial growth. *Journal of Theoretical Biology,* 242(2)**,** 489-501.

Naylor, A. W. & Sell, G. R. (1982). *Linear Operator Theory in Engineering and Science*: Springer-Verlag.

Needham, C. J., Bradford, J. R., Bulpitt, A. J. & Westhead, D. R. (2006). Inference in Bayesian networks. *Nature Biotechnology,* 24(1)**,** 51-53.

Nelson, D. L. & Cox, M. M. (2000). *Lehninger principles of biochemistry* (3rd ed.). New York Worth.

Nielsen, J. (2001). Metabolic engineering. *Applied Microbiology and Biotechnology,* 55(3)**,** 263-283.

Noble, D. (2008). Claude Bernard, the first systems biologist, and the future of physiology. *Experimental Physiology,* 93(1)**,** 16-26.

O'Malley, M. A., Powell, A., Davies, J. F. & Calvert, J. (2008). Knowledge-making distinctions in synthetic biology. *Bioessays,* 30(1)**,** 57-65.

Oliveri, P., Tu, Q. & Davidson, E. H. (2008). Global regulatory logic for specification of an embryonic cell lineage. *Proceedings of the National Academy of Sciences of the United States of America,* 105(16)**,** 5955-5962.

Ozbudak, E. M., Thattai, M., Lim, H. N., Shraiman, B. I. & van Oudenaarden, A. (2004). Multistability in the lactose utilization network of Escherichia coli. *Nature,* 427(6976)**,** 737-740.

Palsson, B. (2006). *Systems biology: properties of reconstructed networks*: Cambridge University Press.

Patel, M. & Crank, M. (2006). Medium and long-term opportunities and risks of the biotechnological production of bulk chemicals from renewable resources. The BREW Project final report. European Commission, GROWTH Programme available at www.chem.uu.nl/brew.

Porter, J. R. (1976). VANLEEUWENHOEK,A - TERCENTENARY OF HIS DISCOVERY OF BACTERIA. *Bacteriological Reviews,* 40(2)**,** 260-269.

Pradhan, D. K. (1978). Theory of Galois Switching Functions. *IEEE Transactions on Computers,* 27(3)**,** 239-248.

Ptashne, M. & Gann, A. (2002). *Genes & Signals* New York: Cold Spring Harbor Laboratory Press.

Ptashne, M. & Gann, A. (2004). *A genetic switch : phage lambda revisited* (3 ed.). New York: Cold Spring Harbor Laboratory Press.

Radatz, J. (1997). *IEEE Standard Dictionary of Electrical and Electronic Terms* (6 ed.): IEEE.

Ragauskas, A. J., Williams, C. K., Davison, B. H., Britovsek, G., Cairney, J., Eckert, C. A., Frederick Jr, W. J., Hallett, J. P., Leak, D. J., Liotta, C. L., Mielenz, J. R., Murphy, R., Templer, R. & Tschaplinski, T. (2006). The path forward for biofuels and biomaterials. *Science,* 311(5760)**,** 484-489.

Ratledge, C. & Kristiansen, B. (2006). *Basic biotechnology* (3 ed.). Cambridge: Cambridge University Press.

Rautenberg, W. (2006). *A Concise Introduction to Mathematical Logic*: Springer Verlag.

Reddy, S. M. (1972). Easily Testable Realizations for Logic Functions. *IEEE Transactions on Computers,* C-21(11)**,** 1183-1188.

Reed, I. S. (1954). A Class of Multiple-Error-Correcting Codes and the Decoding Scheme. *IRE Transactions on Information Theory,* (4)**,** 38-49.

Reed, I. S. (1973). Boolean Difference Calculus and Fault Finding. *SIAM Journal on Applied Mathematics,* 24(1)**,** 134-143.

Reichhardt, C. J. O. & Bassler, K. E. (2007). Canalization and symmetry in Boolean models for genetic regulatory networks. *Journal of Physics A-Mathematical and Theoretical,* 40(16)**,** 4339-4350.

Rodrigo, G. & Jaramillo, A. (2007). Computational design of digital and memory biological devices. *Syst Synth Biol,* 1(4)**,** 183-95.

Rosen, R. (1970). *Dynamical System Theory in Biology - Volume 1: Stability Theory and its Applications*: Wiley-Interscience.

Rosenbrock, H. H. (1970). *Mathematics of dynamical systems*: Nelson.

Russell, P. J. (2006). *Genetics, a molecular approach*. San Francisco Pearson/Benjamin Cummings

Santillan, M. (2008). Bistable behavior in a model of the lac operon in Escherichia coli with variable growth rate. *Biophysical Journal,* 94(6)**,** 2065-2081.

Santillan, M. & Mackey, M. C. (2004a). Influence of catabolite repression and inducer exclusion on the bistable behavior of the lac operon. *Biophysical Journal,* 86(3)**,** 1282-1292.

Santillan, M. & Mackey, M. C. (2004b). Why the lysogenic state of phage lambda is so stable: A mathematical modeling approach. *Biophysical Journal,* 86(1)**,** 75-84.

Santillan, M., Mackey, M. C. & Zeron, E. S. (2007). Origin of bistability in the lac operon. *Biophysical Journal,* 92(11)**,** 3830-3842.

Schlitt, T. & Brazma, A. (2006). Modelling in molecular biology: describing transcription regulatory networks at different scales. *Philosophical Transactions of the Royal Society B-Biological Sciences*, 361(1467)**,** 483-494.

Schlitt, T. & Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8.

Sellers, J. F. F., Hsiao, M. Y. & Bearnson, L. W. (1968). Analyzing errors with Boolean difference. *IEEE Transactions on Computers*, C-17(7)**,** 676-683.

Selzer, P. M., Marhöfer, R. J. & Rohwer, A. (2008). *Applied Bioinformatics: An Introduction*, . Berlin Heidelberg: Springer-Verlag

Setty, Y., Mayo, A. E., Surette, M. G. & Alon, U. (2003). Detailed map of a cis-regulatory input function. *Proceedings of the National Academy of Sciences of the United States of America*, 100(13)**,** 7702-7707.

Shannon, C. E. (1948). Mathematical theory of communication. *Bell System Technical Journal*, 27(3-4).

Shea, M. A. & Ackers, G. K. (1985). The OR Control System of Bacteriophage Lambda - A Physical-Chemical Model for Gene Regulation. *Journal of Molecular Biology*, 181(2)**,** 211-230.

Smolen, P., Baxter, D. A. & Byrne, J. H. (2000). Modeling transcriptional control in gene networks - Methods, recent results, and future directions. *Bulletin of Mathematical Biology*, 62(2)**,** 247-292.

Snoussi, E. & Thomas, R. (1993). Logical Identification of All Steady-States - the Concept of Feedback Loop Characteristic States. *Bulletin of Mathematical Biology*, 55(5)**,** 973-991.

Stamatakis, M. & Mantzaris, N. V. (2009). Comparison of Deterministic and Stochastic Models of the lac Operon Genetic Network. *Biophysical Journal*, 96(3)**,** 887-906.

Stankovic, R. S., Moraga, C. & Astola, J. (2004). Derivatives for multiple-valued functions induced by Galois field and Reed-Muller-Fourier expressions. *In:* Proceedings 34th International Symposium on Multiple-Valued Logic, 2004. ISMVL 2004., 2004. 184-189.

Stankovic, R. S. & Sasao, T. (2001). A discussion on the history of research in arithmetic and Reed-Muller expressions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(9)**,** 1177-1179.

Stephanopoulos, G. (1984). *Chemical process control : an introduction to theory and practice*. London: Prentice-Hall.

Straathof, A. J. J., Panke, S. & Schmid, A. (2002). The production of fine chemicals by biotransformations. *Current Opinion in Biotechnology*, 13(6)**,** 548-556.

Strang, G. (1988). *Linear Algebra and its Applications* (3 ed.): Saunders HBJ.

Thayse, A. (1974a). Differential Calculus for Functions from (GF(p))n into GF(p). *Philips Research Report*, 29(6)**,** 560-586.

Thayse, A. (1974b). New Method for Obtaining the Optimal Taylor Expansions of a Boolean Function. *Electronics Letters*, 10(25-26)**,** 543-544.

Thayse, A. & Davio, M. (1973). Boolean Differential Calculus and its Application to Switching Theory. *IEEE Transactions on Computers*, C-22(4)**,** 409-420.

Thieffry, D., Huerta, A. M., Perez-Rueda, E. & Collado-Vides, J. (1998). From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in Escherichia coli. *Bioessays*, 20(5)**,** 433-440.

Thieffry, D. & Thomas, R. (1995). Dynamical Behavior of Biological Regulatory Networks .2. Immunity Control in Bacteriophage-Lambda. *Bulletin of Mathematical Biology*, 57(2)**,** 277-297.

Thomas, R. (1973). Boolean Formalization of Genetic-Control Circuits. *Journal of Theoretical Biology,* 42(3)**,** 563-585.
Thomas, R. (1991). Regulatory Networks Seen as Asynchronous Automata - a Logical Description. *Journal of Theoretical Biology,* 153(1)**,** 1-23.
Tian, T. H. & Burrage, K. (2005). A mathematical model for genetic regulation of the lactose operon. *Computational Science and Its Applications - Iccsa 2005, Pt 2.* Berlin: Springer-Verlag Berlin.
van Hijum, S., Medema, M. H. & Kuipers, O. P. (2009). Mechanisms and Evolution of Control Logic in Prokaryotic Transcriptional Regulation. *Microbiology and Molecular Biology Reviews,* 73(3)**,** 481-+.
Van Regenmortel, M. H. V. (2004). Reductionism and complexity in molecular biology. *EMBO Reports,* 5(11)**,** 1016-1020.
Vilar, J. M. G. (2006). Modularizing gene regulation. *Molecular Systems Biology,* 2.
Vilar, J. M. G., Guet, C. C. & Leibler, S. (2003). Modeling network dynamics: the lac operon, a case study. *Journal of Cell Biology,* 161(3)**,** 471-476.
Vogelstein, B. & Kinzler, K. W. (2004). Cancer genes and the pathways they control. *Nature Medicine,* 10(8)**,** 789-799.
Wagner, R. (2000). *Transcription Regulation in Prokaryotes*: Oxford University Press.
Wakerly, J. F. (2000). *Digital design : principles and practices* (3 ed.). Upper Saddle River, N.J. ; London Prentice Hall.
Watanabe, Y. & Brayton, R. K. (1993). Heuristic minimization of multiple-valued relations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 12(10)**,** 1458-1472.
Werner, E. (2007). All systems go. *Nature,* 446(7135)**,** 493-494.
Werpy, T. & Petersen, G. (2004). Top Value Added Chemicals from Biomass. US Department of Energy available at www.osti.gov/bridge.
Wesselkamper, T. C. (1978). Divided Difference Methods for Galois Switching Functions. *IEEE Transactions on Computers,* 27(3)**,** 232-238.
Wessels, L. F., van Someren, E. P. & Reinders, M. J. (2001). A comparison of genetic network models. *Pac Symp Biocomput***,** 508-19.
Westerhoff, H. V. (2007). Mathematical and theoretical biology for systems biology, and then ... vice versa. *Journal of Mathematical Biology,* 54(1)**,** 147-150.
Westerhoff, H. V., Winder, C., Messiha, H., Simeonidis, E., Adamczyk, M., Verma, M., Bruggeman, F. J. & Dunn, W. (2009). Systems Biology: The elements and principles of Life. *FEBS Letters,* 583(24)**,** 3882-3890.
Winter, P. C., Fletcher, H. L. & Hickey, G. I. (2002). *Genetics* (2 ed.). Oxford Bios.
Wolkenhauer, O. (2007). Defining Systems Biology: An Engineering Perspective. *Systems Biology, IET,* 1(4)**,** 204-206.
Wolkenhauer, O., Kitano, H. & Cho, K. H. (2003). Systems biology. *IEEE Control Systems Magazine,* 23(4)**,** 38-48.
Wolkenhauer, O. & Ullah, M. (2007). All models are wrong…. some more than others. *In:* Boogerd, F., Bruggeman, F. J., Hofmeyr, J.-H. S. & Westerhoff, H. V. (eds.) *Systems Biology: Philosophical Foundations.* Elsevier.
Wolkenhauer, O., Ullah, M., Kolch, W. & Cho, K. H. (2004). Modeling and simulation of intracellular dynamics: Choosing an appropriate framework. *IEEE Transactions on Nanobioscience,* 3(3)**,** 200-207.
Yuh, C. H., Bolouri, H. & Davidson, E. H. (1998). Genomic cis-regulatory logic: Experimental and computational analysis of a sea urchin gene. *Science,* 279(5358)**,** 1896-1902.

Yunjian, J. & Brayton, R. K. (2000). Don't cares and multi-valued logic network minimization. *In:* Brayton, R. K., ed. IEEE/ACM International Conference on Computer Aided Design, 2000. ICCAD-2000. , 2000. 520-525.

Zika, E., Papatryfon, I., Wolf, O., Gomez-Barbero, M., Stein, A. J. & Bock, A. (2007). Consequences, Opportunities and Challenges of Modern Biotechnology for Europe. European Commission, Joint Research Centre available at www.jrc.ec.europa.eu.