Reprogrammable In Vivo Architecture

Marc Blenkiron

Doctor of Philosophy

Institute for Computing Systems Architecture
School of Informatics
University of Edinburgh

2005

Abstract

The biological cell is the intricate, yet ubiquitous component of life, able to grow, adapt and reproduce. The genetic material contained within a cell encodes information which directs its development and behaviour, and this information is passed down from one generation of cell to the next. One emerging interest, resulting from collaborations between the disciplines of Molecular Biology and Computer Science, is to encode *computational programs*, sets of engineered, information processing instructions, in genetic material, to be executed by living cells.

So far, the large majority of *in vivo* computation research has been based on the detection and conditional manipulation of protein concentrations inside cells, which is the biological method of gene expression. In contrast, this thesis describes how a computational program, encoded in genetic material inside a bacterium, can be triggered by external stimuli to reassemble itself in a directed manner to create a newly arranged computational program.

In order to investigate the potential utility of *in vivo* self-arranging programs, software was designed to explore a search space of candidate computational programs, encoded in genetic material, which are able to rearrange themselves; to simulate these candidates and to evaluate their behaviour against a set of criteria. Rearrangements were facilitated by biological catalysts which can selectively sever and rejoin genetic material in a cooperative manner. Their ability to perform compound operations was found to allow for a general purpose mechanism.

As a proof of concept, one of the candidate computational programs, a two-colour switch which can be set irreversibly through its rearrangement, was encoded in genetic material. Measurements of *in vivo* expression were observed resulting from *in vitro* rearrangement manipulations, to illustrate its operation.

Acknowledgements

I am indebted to my supervisor, D.K. Arvind, whose skill to find and convey promising new areas gave me the opportunity and inspiration to start this work. And, for his encouragement, support and ability to network people with related interests together. I am also indebted to my supervisor, Jamie Davies, for his time and patience in guiding my first steps into Biology. And, for the frequent, enthusiastic and motivating discussions about biological experiments.

Members of the Organogenesis Lab have all offered considerable encouragement and advice. Darren Logan, for introducing many time-saving techniques, for discussions and a great sense of humour. Derina Sweeney, who introduced me to my first wetware experiments and brightened up the lab with her singing. Lydia Michael, whose cheerful encouragement helped keep me going when the latest experiment had to be re-revised. Jane Armstrong, for advice on the practicalities of techniques and somehow knowing where everything in the lab was. And Wen-Chin Lee, for being so helpful, and for gracefully tolerating those occasional attempts to speak in Chinese, no doubt, rather badly.

I am also very grateful for the discussions and a lot of useful criticism given by members of the Speckled Computing Consortium. Also, the members of the Genes and Development Interdisciplinary Group have helped me adsorb many aspects of biological culture and have been keen to express advice. Thanks to Andy Koppe, Ben Schroeder, Jiannis Gregory and Si-Nong Fan whose detailed feedback on the simulation platform and its documentation prompted some considerable improvements. Thanks also to Jean Pierre Lavergne, Kendra White and Matthias Ehrmann who kindly sent a plasmid, which was an important part of my biological experiments, and which had been discontinued by its manufacturer, after their design. And, to Wilson Poon, for his inspiring talk motivating collaborations between biologists and computer scientists, from the perspective of a physicist, and for his subsequent advice.

Finally, and most of all, I would like to thank my parents, Thomas and Lily Blenkiron, for their constant support, love and encouragement. And to Aroosha, for her love, motivation and patience – particularly during the 'finished in about a month' period, which lasted, perhaps, a little longer than that...

Declaration

I declare that this thesis was composed by myself and that the work contained herein is my own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification except as specified.

Some of the work described in this thesis has been previously published.

Arvind, D. K., Blenkiron, M. 2003. *Towards Programmable In Vivo Computation*, Proceedings of the Second Annual Workshop on Non-Silicon Computation, San Diego, USA.

Marc Blenkiron, *September 2005.*

Table of Contents

List of Figures

List of Tables

Biological Abbreviations

AFP	auto-fluorescent protein
amp	ampicillin
aTc	anhydrotetracycline
bp	base pairs
DNA	deoxyribonucleic acid
ds-	double stranded
dNTP	dinucleotide triphosphates
E. coli	*Escherichia coli*
EDTA	ethylenediaminetetraacetic acid
g	grams
HEG	homing endonuclease gene
IPTG	isopropyl-β-D-thiogalactopyranoside
LB	Luria-Bertani
μg	micrograms
mg	milligrams
$MgCl_2$	magnesium chloride
min	minutes
$\mu\ell$	microlitres
mℓ	millilitres
M	molarity
nm	nanometres
PCR	polymerase chain reaction
PEG	polyethylene glycol
RE	restriction endonuclease
RNA	ribonucleic acid
rpm	revolutions per minute
ss-	single stranded
s	seconds
TF	transcription factor
u	standard enzyme units
UV	ultraviolet

Chapter 1

Introduction

1.1 Perspective

In the study of *in vivo* engineered gene networks [Hasty *et al.*, 2002], [Weiss *et al.*, 2003], [Simpson *et al.*, 2004], [Kærn *et al.*, 2004], [Kobayashi *et al.*, 2004] a cell can be viewed as a computational medium [Amos, 2004]. Figure 1.1a illustrates the analogy between the cell and a computer. One form of computer stores state information in volatile memory, transmits information via radio modem, performs computation as directed by a stored program, and all these operations are powered by an energy source, such as a battery. The biological cell stores state information in volatile protein concentrations, transfers information through receptors and transmitters, is powered by the production of its mitochondria and the cell's behaviour is driven by its DNA. The input and output messages of natural cells are communicated using a selection of molecules. Some messages are broadcast using diffusion, others are transferred directly between cellular compartments or touching cells. State, or memory, is held in the concentration levels of degradable proteins and protein templates, some of which can be altered by input messages. The DNA of a cell drives the interpretation of its state and can cause changes in the levels of production of proteins.

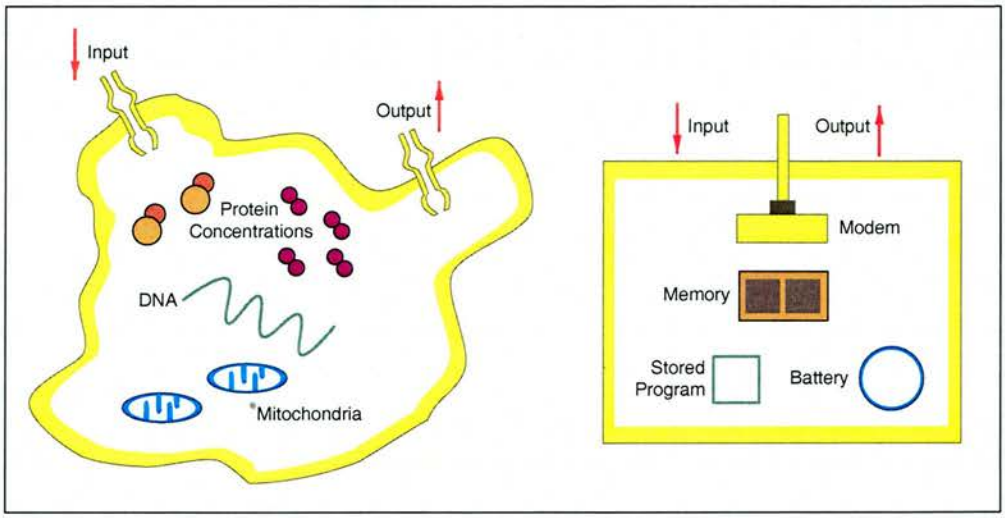


Figure 1.1a: An analogy between a biological cell and a battery-powered computer.

1.2 Motivation

The majority of research into *in vivo* computation has been based on the natural mechanism of *gene regulation*, which, in essence, detects concentrations of proteins and conditionally causes a change in these concentrations. In natural cells, the main function of many proteins is the transfer and processing of information [Bray, 1995]. Such proteins are volatile, in that over time, elements of a host cell's machinery attach to them and destroy them [Podolsky, 1953]. Consequently, if the concentration of a protein is not maintained through continuous production, it will decay. Protein concentrations may therefore be seen as being analogous to the volatile random access memory in silicon-based computers which temporarily stores information.

The flexibility of silicon-based computers is enhanced by using persistent storage, such as magnetic disks which can hold binary-encoded information reliably over extended periods, in the order of decades. Not only can program data be easily stored and replicated, but so can information encoding programs themselves and their configuration. Nature too has its own replicatable, persistent storage mechanism, *viz.* deoxyribose nucleic acid (DNA) which has been used to pass genetic information down through generations of life stretching over billions of years. DNA encodes gene regulation in-

formation. *Genetic programs* are the processes that emerge from complex interactions formed by gene regulation. These programs direct the behaviour and development of a cell. In contrast, a *computational program* is defined in this thesis, as a set of explicit, engineered, information-processing instructions. The encoding of such a set of instructions in DNA is referred to as a *DNA program*. The building blocks of DNA: the nucleotides – of which there are four distinct types, can be chemically linked together in a chain, which can encode a string of quaternary information.

The aim of this thesis is to present the concept of a biological cell being able to reassemble its DNA in response to environmental conditions, thus forming a new DNA program. The thesis demonstrates a potential mechanism using simulations and explores issues of scalability. The plausibility of the approach is confirmed through biological experiments.

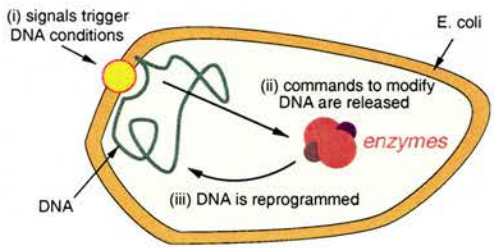


Figure 1.2a: An illustration of a potential mechanism for a DNA-reprogramming operation which is triggered by external stimuli.

The idea of a DNA-reprogramming operation which is triggered by external stimuli is illustrated in Figure 1.2a. One or more external signals, such as a molecule or a light-source, pass through the cell wall and activate specific parts of the cell's DNA. These activations trigger the production of specific DNA-arranging molecules. The cell's DNA is reassembled in a manner which is directed by the type and order of the release of the DNA-arranging molecules.

In general, the rapid realisation of most engineered physical systems is often achieved by arranging standard components together, each having well characterised properties and well specified interfaces. The eventual goal in the case of synthetic biological systems will be to realise generic, *in vivo* reprogrammable behaviour to enable libraries of genetic functional components to be reassembled by living cells in response to external triggers to form new DNA programs, as illustrated in Figure 1.2b. In this case, a signal is generated which causes the production of a molecule which excises a specific component of tagged DNA from a library. Another molecule, generated as a result of a

second signal, opens a chosen destination site for the excised component. Finally, the excised component is inserted into the destination site.

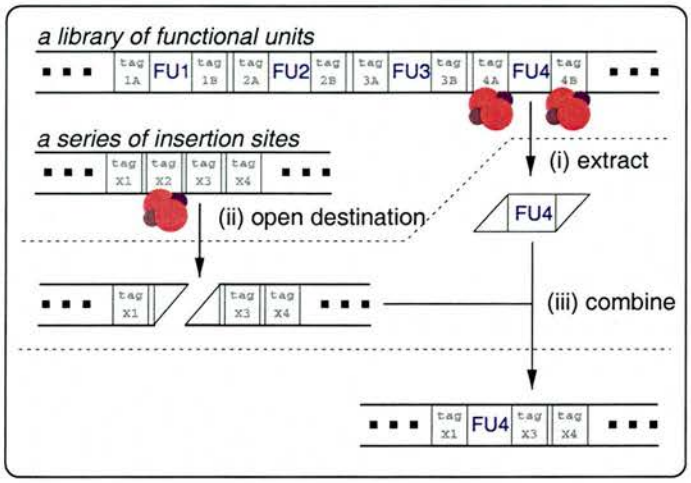


Figure 1.2b: An illustration of a potential, *in vivo* DNA library assembly system.

Signals could either be generated by a person or a machine to reassemble DNA in a predetermined manner, or they could be caused by transitory changes in the environmental conditions which oblige a cell to adapt its DNA in accordance with a DNA program. In this way, environmental events could be permanently recorded in a cell's DNA for inspection at a later time.

The information density of DNA significantly exceeds that of methods used in conventional computers for persistent data storage. Each position in a DNA sequence stores two bits (given four possible nucleotide bases) and each nucleotide base pair requires 63 or 64 atoms (the molecular formulae of G-C and A-T units are $C_{19}H_{22}N_8O_{12}P_2$ and $C_{20}H_{21}N_7O_{12}P_2$ respectively). In contrast, the magnetic films used to store information in the hard disk of a personal computer are typically composed of cobalt, platinum and chromium alloys and require approximately 10^5 atoms to store a single bit of information [Koltsov and Perry, 2004]. Recent research has also demonstrated that 400 cobalt atoms on a platinum substrate can be used to store a single bit at a temperature of 5 Kelvin [Gambardella *et al.*, 2003]. DNA has a clear potential for use as a long-term, high-density storage mechanism in engineered systems.

This potential is indeed considerable when combined with cell-to-cell communica-

tions, already being harnessed in engineered genetic networks [Basu *et al.*, 2005], to realise heterogeneous networks of reprogrammable cells.

1.3 Approach

The first investigation in this thesis was to query the potential of a class of biological catalysts to rearrange a DNA program inside a bacterium, in a directed manner, driven by external stimuli. To address this goal, a notation was developed to describe DNA programs, their execution and rearrangement capabilities. Building on this notation, a simulator was developed to automate the analysis of DNA programs written in this notation. The simulator can dynamically map between DNA programs (in notation) and a model of networks of interacting genetic components with stochastic reactions (suitable for simulation) and thus supports the simulation of DNA programs which reprogram themselves *during* their own execution. Building on the simulator, a search tool was created to generate DNA programs, composed from a small database of sites and reaction constants, which fulfilled a set of specified rule-based behavioural criteria. Owing to the complexity of the simulated operation of some self-arranging DNA programs, the generator was designed both to *justify* its results and provide concise descriptions of simulated DNA arrangement events over time which can be referenced against output RNA measurements.

The abilities of the generator to design DNA programs and the simulator to execute them, were verified by testing the generator against the behaviour of small, characterised regulatory networks. For example, the generator was instructed to produce an externally-settable toggle switch using two different protein input signals and a reporter gene. This resulted in the generation of a mutually-inhibitory network similar to a DNA program created by Gardener *et al.* in 2000. A more complex query to the generator required a DNA program to act as a two-colour memory latch which can be set by a sugar input signal. The resulting DNA program with a memory latch is similar to that used by bacteriophage λ [Ptashne, 2004]. This example also demonstrated the ability of the generator to make use of intermediatery protein signals, which were not used as inputs or outputs, but instead to retain internal state.

Next, the generator was instructed to find DNA switches which could remember their state, but without relying on protein concentrations to accomplish this. A selection of

enzymes with different methods of cleaving DNA were assessed in these experiments. It was found that the recognition sequences and cohesive ends produced by enzymes could direct the different types of DNA arrangement operations that contributed to the operation of a DNA switch. The generator was then asked to place a successful DNA switch under the control of gene regulation to demonstrate the operation of a complete, model rearrangement system, *in silice*.

It was also important to demonstrate whether or not such a potential DNA arrangement system could be computationally scalable. That is, whether combinations of DNA arrangement operations could be applied to a DNA program to *compound* their effects and increase the space of potential rearrangements. A DNA program was created to demonstrate a general purpose DNA rearrangement mechanism. It was shown how four different enzymes can be used, at two time points, to effect directed rearrangements causing the production of eight different DNA programs with distinguishable input/output behaviours. Also, when allowing the excision of genes to take place, a state space of 36 DNA programs could be attained, in principle, using only four enzymes.

Finally, one of the example DNA programs created by the software, the DNA switch, which could demonstrate an arrangement operation *in silice*, was implemented in wetware to assess its biological plausibility. Rotational operations were performed *in vitro* and measurements were made *in vivo*. In addition to demonstrating the principle of a rearrangement operation, a number of important characteristics of the DNA switch were assessed. It was found that the state of the DNA switch is stable across generations of bacteria and that its state is clearly distinguishable. Also, once set, using an enzyme with an asymmetric recognition sequence, the state of the DNA switch is irreversible, even if subjected later to the same enzyme.

1.4 Thesis Outline

In the next section, biological terms and concepts are explains for a reader more familiar with computer science (Section 1.5). The following section introduces the laboratory techniques that were used in the biological experiments (Section 1.6). In the final section of this chapter, a symbolic notation is described which represents the properties of a DNA program and its interactions with aspects of a biological cell's machinery.

This notation is used to illustrate key aspects of gene regulation in the context of this work (Section 1.7).

The following chapter describes related work in the creation of synthetic DNA programs and the contributions towards our ability to design successful DNA programs (Chapter 2).

An *exploratory* software program is presented which can search for DNA program encodings based on the symbolic notation. Its search strategy, which can be directed by a set of behavioural criteria, is also described. Simulation *platform* requirements are stated which result from wishing to predict the behaviours of points in a reprogrammable DNA search space. The underlying simulation *model* is then defined which is utilised by the exploratory program to predict the behavioural outcomes of a DNA program (Chapter 3).

The results of using the exploratory program to search for DNA programs able to act as switches, storing their states in both protein concentrations and in DNA itself, are described and compared. The potential scalability of reassembling DNA programs is also demonstrated through simulation (Chapter 4).

One of the DNA programs found by the exploratory program, a two-colour, DNA switch with a rotatable mid-section, was implemented in wetware. The stages involved in its construction are described as are the protocols used to cause the switch to be set and to test the switch's robustness. Next, the mechanism of the DNA switch is demonstrated, as are aspects of its operation that are important for its potential use as a computational unit: stability across generations of bacteria, ease of distinguishing its state and irreversibility of a state-setting operation (Chapter 5).

The consequences and limitations of this work are discussed in the context of related research and future work is presented (Chapter 6).

The first appendix chapter provides additional details of how the software presented in this thesis can be used and how it works (Appendix A). Supplementary information relating to the biological experiments is provided in Appendix B. And finally, Appendix C contains a previously published paper that is relevant to this thesis.

1.5 DNA

This section explains some of the biological terms and concepts used in this thesis for a reader more familiar with computer science. A more detailed introduction can be obtained by consulting resources introducing molecular cell biology [Alberts *et al.*, 2002], [Pollard and Earnshaw, 2004].

1.5.1 Reading

DNA encodes the developmental and behavioural behaviour of cells and is inherited from one generation of cell to the next. A single strand of DNA, *ssDNA*, is composed of a string having four possible types of *base*: adenine, thymine, cytosine and guanine. Certain base patterns encode specific sites that determine how DNA is interpreted by a cell.

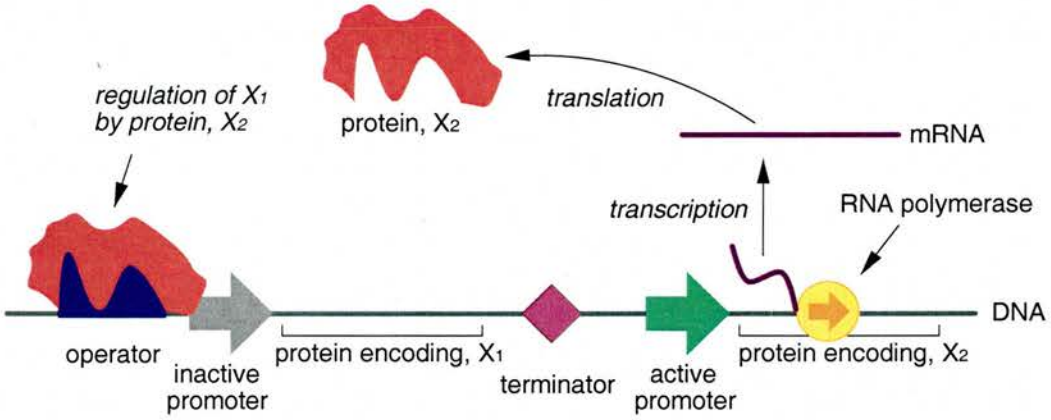


Figure 1.5.1: A simple model of transcription, translation and gene regulation.

A model of how a bacterium interprets its DNA is shown in Figure 1.5.1. Reading commences when an enzyme called *RNA polymerase* attaches stochastically to a *promoter* site that is encoded in the DNA. The promoter sites specify the forward direction in which RNA polymerase should continue. When a site that encodes a specific protein is read, a corresponding template for a protein, *mRNA*, will be produced. This process is called *transcription*. A cell can *translate* mRNA templates into actual proteins. The frequency of reading from a particular promoter can be regulated by *operator* sites. If specific proteins in the cell attach stochastically to these sites, this can increase or decrease the probability that a nearby promoter is read from. Furthermore, some operator

sites exhibit *cooperative binding* which changes the affinity of one kind of protein for a site, if another protein binds to it. Positive cooperative binding is exhibited when the binding of one protein increases the affinity of another protein to bind. Conversely, a negative cooperative binding will cause the reduction in affinity for one protein, when another protein is bound. Reading can be halted when a *termination* site is encountered. When a cell contains more than one RNA polymerase molecule, each molecule can read a different part of the DNA at the same time. The process described here, where DNA is transcribed into RNA, which, in turn, is translated into proteins, represents the standard flow of information in a cell [Crick, 1958]. A segment of DNA that contains the encoding of a protein and regulatory sites is called a *gene*. A graph that is composed of the connections formed by the regulatory effects of proteins on genes is called a *genetic network*.

1.5.2 Complementarity

The DNA in a cell is usually double stranded DNA, *dsDNA*, and is composed of two *ssDNA* strands, where each base on one strand is bound to a *complementary* base on the other: Adenine and thymine are complementary, as are cytosine and guanine. RNA has a similar system, except that thymine is replaced with uracil. Two single nucleic acid strands, which are composed of multiple bases, are complementary if, when aligned, every pair of bases (called a *base pair*) is complementary. Double-stranded DNA (and RNA) is composed of two complementary strands.

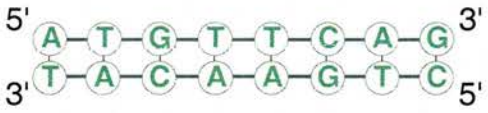


Figure 1.5.2: An illustration of double-stranded DNA. The base pairs of the two complementary sequences are connected with hydrogen bonds (vertical lines) and the sequential bases in each strand of DNA are connected with covalent bonds (horizontal lines). Each strand has a beginning and an end and these are marked with *5'* and *3'* respectively.

Owing to the physical nature of the double-stranded DNA backbone, each strand has a beginning, called 5' (five prime) and an end, called 3' (three prime). Two strands, that are complementary when oriented in opposite directions, can anneal – form hydrogen bonds between complementary bases, whether they are both DNA, RNA, or one of each (Figure 1.5.2).

1.5.3 Anti-Sense Transcripts

An anti-sense transcript is produced when a gene, in double stranded DNA, is read backwards (*anti-sense* direction). The resulting mRNA molecule encodes the complimentary bases that a forward transcript would. Consequently, an anti-sense transcript can bind with a normal (*sense*) transcript and does so with high affinity, preventing the sense transcript from acting as a template for protein production (Figure 1.5.3).

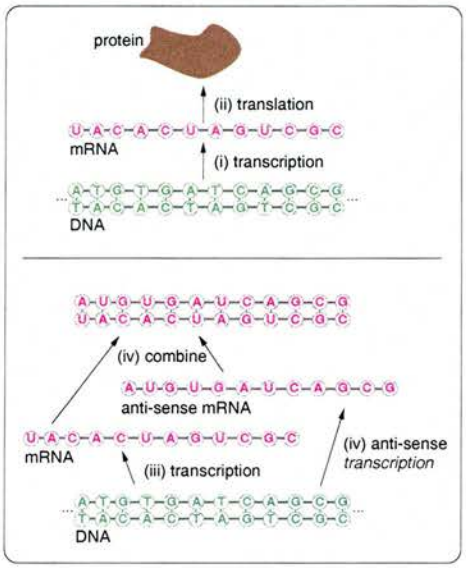


Figure 1.5.3: An illustration of how protein production can be hindered by anti-sense transcripts. (i) mRNA molecules being transcribed from DNA when read in the forward (sense) direction by RNA polymerase. The mRNA bases correspond the the DNA bases with thymine, T, being replaced by uracil, U. (ii) Ribosome molecules translate the mRNA template into proteins, unhindered by anti-sense transcripts. (iii) mRNA being transcribed in the sense direction. (iv) mRNA being transcribed in the anti-sense direction creating a molecule that is complementary to the sense mRNA. (v) The sense and anti-sense mRNA have a strong affinity for each other and bind together, preventing ribosomes from translating them.

Therefore, anti-sense transcripts of a protein have the effect of reducing its production by interfering with its normal mRNA transcripts.

1.5.4 Cohesive Ends

Cohesive ends are single strands of DNA which extend from double-stranded DNA (dsDNA). An example of DNA with cohesive ends is shown in Figure 1.5.4a.

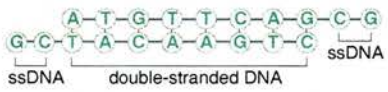


Figure 1.5.4a: An illustration of cohesive ends. Two single-strands extend out from the central dsDNA molecule.

Some enzymes, called *restriction endonucleases*, can recognise special sites in a DNA sequence and *digest* – cut – the DNA sequence on one or both strands. Cohesive ends are formed when dsDNA is digested on both strands at different places (Figure 1.5.4b).

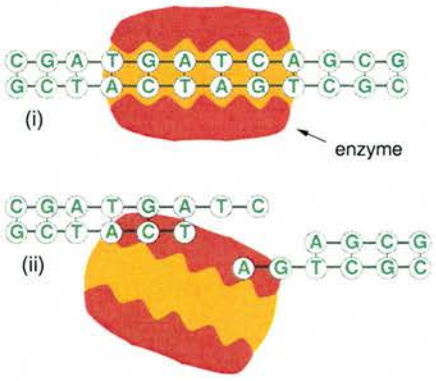


Figure 1.5.4b: An illustration of DNA being digested to produce cohesive ends. (i) An enzyme attaches to dsDNA. (ii) The DNA is digested and two single strands of DNA are produced where the DNA was broken.

The resulting single-stranded DNA (ssDNA) extensions may anneal to other extensions where the bases in the ends are complimentary. The process of annealing connects the complementary bases across the two strands of DNA, but breaks each sequence still remain. An enzyme called *DNA ligase* can repair breaks in each ssDNA sequence by forming covalent bonds between each break. Cohesive ends that are able to anneal to each other are described as *compatible*. When a restriction endonuclease digests DNA on both strands at the same location, this results in *blunt ends*, which do not contain single-stranded base extensions. Blunt ends may also be ligated together, but with a lower efficiency than that of cohesive ends.

1.5.5 Plasmids

One of the requirements for the replication of DNA inside a bacterium is that it is in plasmid form. A DNA plasmid is composed of dsDNA where each string is connected to itself in a loop (i.e., the strings have no end-points).

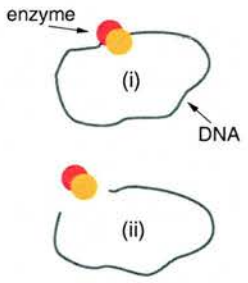


Figure 1.5.5: An illustration of a DNA plasmid being linearised. (i) An enzyme attaches to a plasmid and digests a section of its DNA. (ii) The plasmid is linearised and is no longer in a loop.

When a dsDNA plasmid is digested on both strands, this results in dsDNA which is no longer circular and is said to have been linearised (Figure 1.5.5). Conversely, when cohesive ends (Section 1.5.4) of the same linear DNA molecule are joined together in a loop the DNA forms a plasmid and is said to have been circularised.

1.5.6 Ambiguity

The information encoded by DNA is represented by a string of four types of base and some enzymes recognise particular strings of bases (e.g. "GCTAGC") on DNA which causes the enzymes to react with them. Of these enzymes, some recognise strings with base choices (e.g. "CCT[AVTVGVC]AGC"). Such choices are termed *ambiguity* and a standard nomenclature is used to represent these concisely [NCIUB, 1985] as shown in Table 1.5.6.

R ← G ∨ A	B ← C ∨ G ∨ T
Y ← C ∨ T	D ← A ∨ G ∨ T
M ← A ∨ C	H ← A ∨ C ∨ T
K ← G ∨ T	V ← A ∨ C ∨ G
S ← G ∨ C	N ← A ∨ C ∨ G ∨ T
W ← A ∨ T	

Table 1.5.6: Recognition sequence ambiguity codes. Each code can represent a number of bases. For example, the ambiguous sequence, "ABC", could represent "ACC", "AGC" or "ATC".

1.6 Laboratory Techniques

This section introduces the standard laboratory techniques that were used during the experiments in Chapter 5. It is intended to aid the accessibility of the biological protocols to a reader who is unfamiliar with them. Detailed descriptions of the standard protocols used in this thesis are described in the Appendix (Section B.1). For an in-depth discussion of a range of molecular cloning techniques, the reader is directed to a standard laboratory manual [Sambrook and Russel *et al.*, 2001].

1.6.1 Cloning

The bacterium, *E. coli*, has been well-studied and can multiply rapidly. One use of *E. Coli* is to clone DNA, i.e. to make many copies of an engineered DNA sequence. Another use of *E. Coli* is to express the genes encoded by a engineered DNA sequence and to collect or analyse the proteins that are synthesised.

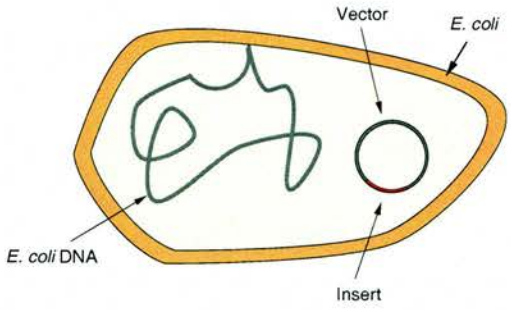


Figure 1.6.1a: An illustration of a plasmid inside an *E. coli* cell. The circular plasmid is composed of a vector and an insert.

A first consideration before one can place an engineered DNA sequence into *E. Coli*, is that in order to be able to replicate the DNA, it must be in a circular form (Section 1.5.5) and have an origin of replication [Messer, 2002]. A number of bacterial viruses exploit the DNA replication mechanism of *E. coli* by carrying their genetic material in plasmids which are copied by an infected *E. coli* cell each time it grows and divides. By linearising the plasmid of a bacterial virus, one can then insert a compatible sequence of DNA at the point of digestion and ligate it into place. In the context of cloning, the viral plasmid is usually referred to as a *vector* and the additional DNA sequence, as an *insert* (Figure 1.6.1a). When an *E. Coli* cell replicates the plasmid, it will also replicate the DNA sequence that was inserted into it. Many vectors are commercially available and some have additional properties that aim to help the cloning process. For example, the vector, *pUC18* [Yanisch-Perron *et al.*, 1985], is a plasmid which contains a cloning region – a short sequence of DNA that encodes restrictions sites for twenty commercially available restriction endonucleases. This is convenient as it increases the range of possible cohesive ends (Section 1.5.4) that can be generated to which one can ligate the compatible ends of an insert.

E. Coli cells do not readily accept the introduction of vectors as their cell membranes act as barriers to these molecules. In order to make the cells *competent* – able to accept the vectors – a commonly used method is that called *heat shocking*. During this process, *E. Coli* cells are subjected to a short temperature pulse of around 42°C in a calcium chloride solution that contains the vectors. On the one hand, this creates a thermal gradient from the solution into the cells. On the other hand, the calcium chloride ions reduce the repulsion of the negatively charged DNA vector to the negatively charged cell membrane.

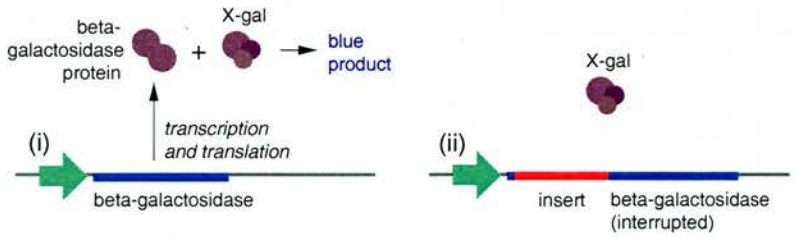


Figure 1.6.1b: An illustration of how blue/white screening works. (i) The gene coding for beta-galactosidase is read and the resulting protein reacts with X-gal to produce a blue product. (ii) The gene is interrupted by an insert and as no beta-galactosidase protein is synthesised, a blue product is not produced.

The cells that accepted the vector are described as having been *transfected*. Usually, only a minority of the cells will have been transfected and so it is useful to isolate these, from the others. Some vectors, such as *pUC18*, contain a gene that confers resistance to the antibiotic, ampicillin. By growing cells in a solution that contains ampicillin, those with the antibiotic resistance are more likely to survive, and therefore to make more copies of themselves (including the vector), as they grow and multiply. This kind of situation, where cells with the vector have a better chance of survival, can be referred to as *selection*. Another method used to isolate cells which contain the vector and insert is called *blue/white screening*. The cloning region of some plasmids, including *pUC18*, is placed at the start of a gene which codes for the protein, beta-galactosidase. If an insert of sufficient length is placed into the cloning region, this disrupts the production of this protein. Beta-galactosidase can metabolise the modified sugar, X-gal in the presence of IPTG and in doing so produces a blue product. Therefore, by growing cells on a Petri dish in a medium that contains X-gal and IPTG, those colonies which contain the vector, but not the insert, will turn blue and those with the insert will remain white (Figure 1.6.1b). One can then use a sterile rod to scrape some white cells from the dish and then place them in a medium in which they can grow and multiply. This process of choosing cells is called *picking*, and the process of adding of cells to a sterile medium can be referred to as *inoculation*.

Analogous to the making of a backup copy of valuable computer data, one can make stock solutions of bacteria containing engineered DNA that can be stored reliably for a decade or longer. One method of storage involves adding glycerol to a vial of cells and then placing them in a freezer at a temperature of -70°C or lower. This method

provides the convenience that one can scrape the top of a vial of frozen cells with a sterile rod, and use this to inoculate a medium with suitable nutrients, in which the cells can then start to grow.

1.6.2 Analysis

To extract vectors containing inserts from a colony of cells one can use a technique called a *mini-prep*. During this process, the cell membranes are broken open and the plasmid DNA is isolated by a series of chemical and physical separation steps.

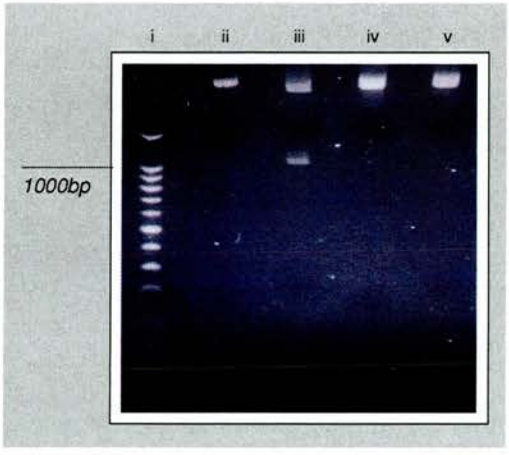


Figure 1.6.2a: A gel electrophoresis result visualised with a UV light source. This gel contains five vertical lanes (labelled *i*-*v*): The lane, *i*, contains a DNA ladder which is composed of DNA fragments of known sizes. The other four lanes contain experimental DNA. This gel indicates that the lane, *iii*, contains DNA which has a length of slightly more than 1000bp (base pairs) but that experimental DNA in the other lanes do not contain DNA of this length.

A number of techniques can be used to investigate or to verify the composition of a plasmid. One method, called *gel electrophoresis*, makes use of the negative charge that DNA carries. By placing DNA in wells in a slab of gel, and by applying a current across the gel, the DNA will move towards the positively-charged anode. Linear DNA molecules travel at a speed that is inversely proportional to the logarithm of their size in base pairs. Ethidium bromide (EtBr) can bind to DNA and glows under UV light, so by adding EtBr to the gel, this allows the visualisation of the DNA with a UV source. Therefore, by placing DNA fragments of known sizes, called a *ladder*, into one well, and the DNA under investigation into another, one can estimate the size of this DNA (Figure 1.6.2a). One can also digest DNA with restriction endonucleases at expected restriction sites, and perform gel electrophoresis to verify that the sizes of the resulting fragments of DNA are as expected. Gel electrophoresis can also be used to separate strands of DNA that have different sizes. For example, if one uses restriction endonucleases to extract a section of DNA, such as a gene, from a plasmid, one can separate the extract from the rest of the plasmid by gel electrophoresis. One can then cut out

the part of the gel containing the extract with a scalpel and recover the DNA from the gel by applying an *in vitro* chemical process.

Another method that can be used to verify the composition of a plasmid is the *Polymerase chain reaction* technique (PCR) which *amplifies* – makes copies of – DNA that exists between two specified sequences (Figure 1.6.2b). One firstly chooses short sequences of single-stranded DNA, called *oligonucleotides*, which match the expected beginning of the DNA sequence under investigation and the complementary sequence of its end. These oligonucleotides, which are referred to as *primers*, are added to a tube containing the DNA under investigation and *DNA polymerase* – which can copy DNA. The tube is then subjected to a cycle of temperatures. Firstly, the tube is heated to cause the separation of each dsDNA molecule into two ssDNA molecules. Next, the temperature is lowered to allow the *annealing* – the forming of hydrogen bonds – between the primers which are complementary to regions of the DNA under investigation. In the third stage, the temperature is increased to allow the DNA polymerase to extend the primers that have annealed to the ssDNA to form dsDNA molecules.

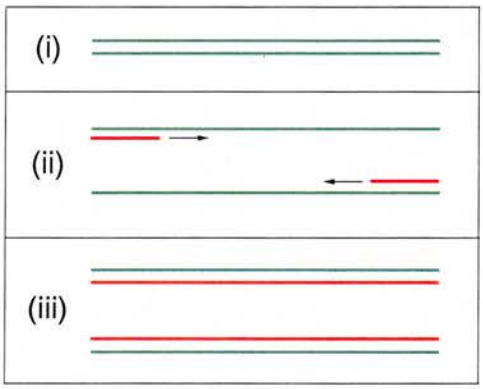


Figure 1.6.2b: An illustration of how PCR amplifies DNA. (i) Double-stranded DNA. (ii) The two strands of dsDNA are separated and then primers attach to complementary regions at the end of each strand. These primers are extended, in the direction of the arrows, by DNA polymerase. (iii) The extending of the primers results in two copies of the original double-stranded DNA.

The rate of copying is exponential with respect to the number of cycles: in one cycle, each ssDNA molecule can be used to make a dsDNA molecule and in the next cycle *both* ssDNA strands of this dsDNA molecule can be used as templates. If the primers do not match the DNA and are not able to anneal, then the DNA cannot be amplified. Therefore, one can verify the existence and size of DNA between two sequences by subjecting it to PCR and then using the result in a gel electrophoresis test. Amplification can also be used to create a large number of copies of a sequence of DNA *in vitro*.

1.7 Notation

1.7.1 Purpose

A symbolic notation was developed that describes the units composing a *DNA program* (Section 1.7.2), which describes a set of genetic regulatory components, and the relationships between these units (Section 1.7.3). It is intended to aid both the accessibility of the biological aspects of this thesis to a computer science audience and the accessibility of the simulation aspects to a biological audience.

The format of the notation has been inspired by the biological convention of a gene-line. The composition of a DNA program can be directly interpreted as a string of symbols which allows symbolic manipulation (Section 3.5) and the application of search algorithms (Section 3.4). For a DNA program composed of constituent units from a database of behavioural properties, and given the relationship between them, the resulting genetic network can be simulated directly (Section 3.7). DNA programs that cause their own rearrangement to form different ones can also be expressed in this notation.

To reduce complexity and aid simulation efficiency (Section 3.6), DNA units can be described at two levels of abstraction: *sites* with specified behavioural characteristics and the *bases* which implicitly compose sites, the latter having inferred behavioural characteristics.

1.7.2 Tape Composition

Prokaryotic cells running DNA programs may be seen as instances of a Turing Machine (TM) [Turing, 1936] that is non-deterministic [Cook, 1971] and makes probabilistic choices. Each RNA polymerase molecule is analogous to a TM head and the DNA program corresponds to a TM tape. The interactions of RNA polymerase molecules with DNA sites are influenced by stochastic events therefore these interactions are not deterministic [McAdams and Arkin, 1999]. A symbolic notation will now be presented that is used to describe the composition of DNA programs [Arvind and Blenkiron, 2003]. This notation has both descriptive and functional rôles in this thesis. In descriptive terms, the notation is used to present a computational perspective of DNA program execution while introducing mechanisms of gene regulation relevant to this work; in

functional terms, the notation can be read to form a gene network which can then be simulated to predict its possible behaviours (Section 3.7.2). The rearrangement of a DNA program can be readily accomplished by the manipulation of the linear string of symbols that encode a DNA program in this notation. Models of genetic networks can be formed for stochastic simulations, by converting a string of symbols into a network of nodes: Each symbol represents a genetic site, e.g. an operator. The possible interactions between molecules and genetic sites are also encoded in the notation. One can therefore form a network of genetic sites where each node represents a symbol in the string (Sections 3.7.2.1 and 3.7.3) and each connection between the nodes (Section 3.7.2.2) represents a potential interaction, e.g. a regulatory influence, or a potential route for a molecule, e.g. the attachment of a protein to an operator site. During a simulation, molecules are passed between nodes over the connections with probabilities and delays that are calculated by consulting the properties of the connections and the states of the nodes. For example, a promoter node may have its affinity for RNA polymerase reduced because a nearby operator site is causing its repression and therefore the probability of an RNA polymerase molecule attaching to the promoter is reduced.

Furthermore, nucleotide base sequences can be encoded in the notation to dictate the possible DNA program rearrangement possibilities, in the presence of specific, DNA-arranging enzymes. Therefore, this notation supports the rearrangement of DNA programs being driven by their own simulated execution and that new genetic networks can emerge from the re-formed DNA programs.

To illustrate the execution of part of some DNA, consider a program as though it were laid out on a tape[1], $\Gamma = \{ P_{id} \mid T_{id} \mid O_{id} \mid X_{id} \mid R_{id} \}^*$, where the components of such a tape are described in Table 1.7.2a.

Symbol	Meaning
P_{id}	promoter
T_{id}	terminator
O_{id}	operator (regulator)
X_{id}	protein encoding
R_{id}	recognition sequence

Table 1.7.2a: Sites that can be used to compose a DNA program.

[1]When a superscript asterisk is used in an equation, it specifies that the symbol or the section in braces that it follows can be repeated any number of times, including none. When symbols are listed inside braces and separated by a bar ('|'), this indicates that any one of the symbols can be chosen.

The tape, of type, Γ, is an ordered list of components made up of promoters, P_{id}, terminators, T_{id}, operators, O_{id}, protein encodings, X_{id}, and special sequences that are recognised by enzymes that can rearrange DNA, R_{id}. Each component is identified by a name, *id*. Any two components having the same type and name are considered to have the same nucleotide base sequence. Promoters allow transcription to begin; terminators can halt transcription; operators can alter the probability of transcription beginning from a promoter; when a protein encoding is transcribed, mRNA is produced; recognition sequences represent a nucleotide sequence that is recognised by DNA-modifying enzymes. Components are assumed to point to the right (upper strand) unless there is an arrow above pointing to the left (lower strand). For clarity, a convention has been adopted where site categories of a DNA program are encoded italicised and in blue to distinguish them from other components, such as proteins with the same subscripted identifiers. The example Figure 1.7.2b illustrates a small part of the genetic code of a bacterial virus, phage λ, in this notation.

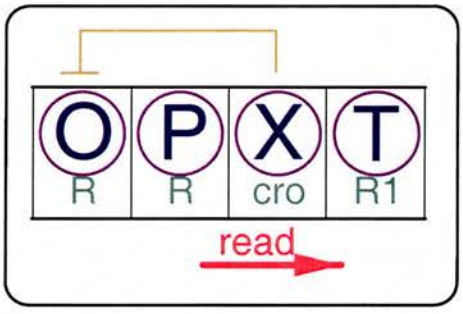


Figure 1.7.2b: A graphical representation of the DNA program, O_R P_R X_{cro} T_{R1}.

An operator, O_R sits beside a promoter, P_R from which RNA polymerase can attach and cause the transcription of protein encoding, X_{cro}. RNA polymerase is halted by terminator, T_{R1}. A brown line shows that the Cro protein, produced by translating mRNA from the transcription of X_{cro}, is able to attach to the operator site, O_R. The possible route of RNA polymerases is shown by a red arrow. The promoter, P_R, and terminator, T_{R1}, mark the start and the end of transcription, respectively. During transcription, a protein encoding, X_{cro}, produces an mRNA template from which the protein Cro is synthesised. This can attach itself to the operator site, O_R, which inhibits, or *downregulates*, the promoter, P_R. This process hinders further transcription. The products of mRNA and Cro are subject to decay and thus the negative feedback loop regulates the amount of Cro.

1.7.3 Unit Relationships

So far, only the layout of the sites of a DNA program have been introduced; the relationships between DNA, the actions of RNA polymerases, proteins and transcripts have been assumed. This section describes properties of RNA polymerases, proteins and transcripts as they relate to each other and the DNA sites.

An RNA polymerase of type, Φ, describes the behaviour of a transcribing enzyme (Equation 1.7.3a). The likelihood of an RNA polymerase molecule attaching to different types of promoter is encoded as a set of pairs ($\mathbf{P}_{\text{id}}$, $\Re^+$) of promoter type and likelihood value, where $\Re^+$ is a positive number, specific to each pair. Termination effects are encoded similarly as sets of pairs, ($\mathbf{T}_{\text{id}}$, $\Re^+$). The regulatory influences of operators are encoded in sets of quartets, ($\mathbf{O}_{\text{id}}$, $\mathbf{M}_{\text{id}}$, $\mathbf{P}_{\text{id}}$, $\Re$), each set encoding the following: the specific positive or negative regulatory effect, $\Re$, on promoter, $\mathbf{P}_{\text{id}}$, when a protein, $\mathbf{M}_{\text{id}}$, is attached to a nearby operator site, $\mathbf{O}_{\text{id}}$. Proteins derive their identifier from the protein encoding from which they were produced and protein $\mathbf{M}_{\emptyset}$ is used to represent the absence of a protein attached to an operator. For example, transcripts from an $\mathbf{X}_{\text{cro}}$ site are translated to produce Cro proteins of type, $\mathbf{M}_{\text{cro}}$. Finally, the size of the RNA polymerase protein, measured as the number of encoding nucleotide base pairs, which map in triples to the amino acids composing the protein, is stated (κ_{bp}).

$$\Phi = \{(\mathbf{P}_{\text{id}}, \Re^+)\}^*, \{(\mathbf{T}_{\text{id}}, \Re^+)\}^*, \{(\mathbf{O}_{\text{id}}, \mathbf{M}_{\text{id}}, \mathbf{P}_{\text{id}}, \Re)\}^*, \kappa_{bp} \tag{1.7.3a}$$

The interactions of an *E. coli* RNA polymerase molecule with a subset of phage λ sites are expressed in Equation 1.7.3b using arbitrary example constants.

$$\Phi_{\text{example}} = (\mathbf{P}_{\text{R}}, 0.6), (\mathbf{T}_{\text{R1}}, 0.5), (\mathbf{T}_{\text{R2}}, 1.0), (\mathbf{O}_{\text{R}}, \mathbf{M}_{\text{cro}}, \mathbf{P}_{\text{R}}, -0.95) \tag{1.7.3b}$$

This RNA polymerase has a strong affinity for $\mathbf{P}_{\text{R}}$ (0.6) and therefore has a high likelihood of starting transcription from a $\mathbf{P}_{\text{R}}$ site. But when an $\mathbf{M}_{\text{cro}}$ protein is attached to a nearby $\mathbf{O}_{\text{R}}$ operator, this promoter will be effectively blocked [Ptashne *et al.*, 1980], thus lowering this chance, by multiplying the RNA polymerase's affinity for the promoter by 0.05 (-0.95+1). The terminator, $\mathbf{T}_{\text{R1}}$, is effective half of the time [Rosenberg *et al.*, 1978] (0.5), whereas the terminator, $\mathbf{T}_{\text{R2}}$, is fully effective (1.0). A DNA program which is composed of sites from Equation 1.7.3b and represents a simplified short tape of another part of phage λ is shown in Figure 1.7.3c.

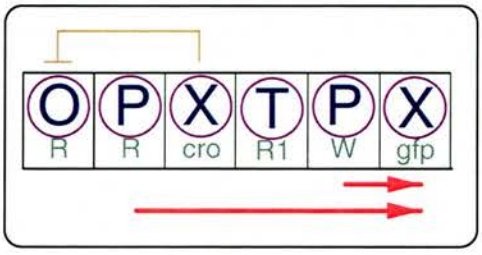


Figure 1.7.3c: A graphical representation of the DNA program, O_R P_R X_{cro} T_{R1} X_{cII} T_{R2}.

An operator O_R sits upstream of a promoter, P_R that can cause RNA polymerase to transcribe from X_{cro}. The route of RNA polymerases is partially halted by the terminator, T_{R1} and those RNA polymerases continuing past this promoter transcribe X_{cII}. In this simple example, no relationships have been defined for X_{cII} and so its sole purpose here is to encode an output protein signal. All RNA polymerases passing through terminator, T_{R2} are stopped. Therefore, on average, for half of the transcriptions on this tape, X_{cro} alone will produce a transcript; otherwise, a transcript from both X_{cro} and X_{cII} will be produced.

The properties of proteins, other than RNA polymerases, of type M, are encoded in four parts (Equation 1.7.3d). Firstly, the decay rate of the protein is stated when unattached to any DNA site (δ_{un}). Secondly, the affinities of a protein for operator sites are stated. When a protein is attached to an operator site, part of its surface is hidden from the machinery of a cell that is responsible for causing the decay of proteins. Consequently, the decay rate, per unit time, of the protein when attached, is also encoded to form a set of triples: (O_{id}, $\Re^+$, δ_{at}). Thirdly, the DNA-arranging properties of the protein, if any, are described in sets of quintets. Each set describes a sequence of nucleotide base pairs (R_{id}) recognised by the protein which may be ambiguous (Section 1.5.6). Base pair positions relative to the start of the sequence where the DNA is spliced on upper and lower strands are encoded (Z_u and Z_l) along with the relative effectiveness of this protein to modify DNA ($\Re^+$). Some DNA modifying enzymes rejoin spliced DNA ends and others leave DNA broken. This information is encoded in a flag ($\{T|F\}$). Finally, the size of the protein is encoded (κ_{bp}). This information provides an indication of the time taken to transcribe and translate this protein and therefore the cost of its synthesis.

$$M = \delta_{un}, \{(O_{id}, \Re^+, \delta_{at})\}^*, \{(R_{id}, Z_u, Z_l, \Re^+, \{T|F\})\}^*, \kappa_{bp} \tag{1.7.3d}$$

A set of relationships describing the interactions in this notation system may be specified in general by, $\{\Gamma^*, \Phi^*, M^*\}$. That is, a collection of tape configurations, and the relationships between these and RNA polymerases and proteins.

Summary

This chapter has presented the motivations for developing synthetic, self-arranging programs and the biological context that will be used in this thesis. The next chapter will survey DNA programs that demonstrate a range of functions, and different methods of DNA arrangement will be described and compared. Finally, current methods that assist in the creation and understanding of successful *de novo* DNA programs will be presented.

Chapter 2

Related Work

2.1 Overview

Synthetic biology [Benner and Sismour, 2005], [Pennisi, 2005] is an emerging discipline which seeks to design and assemble molecular components which are incorporated into biological cells [McDaniel and Weiss, 2005], [Endy, 2005]. On the one hand, this provides an opportunity to investigate biological systems by adding new, or replacement components to cells and then analysing their behaviour, rather than by removing or otherwise silencing the operation of existing natural components. On the other hand, some synthetic biologists seek to design and engineer the functionality of computational devices by assembling biological components able to sense their environmental conditions, process information and generate output effects.

Collaboration between biologists and computer scientists has produced an array of methods aiding the analysis of experimental biological data. Techniques in modelling and simulation are contributing towards our understanding of natural biological systems [Jong, 2002]. As the potential of engineering technologies enabling the modification of genetic material increases, so does our opportunity to compose *de novo* biological devices. One particular route of interest, investigated in this thesis, is to explore the potential of biological cells as media for engineered, biological, computational devices.

2.2 Information Storage and Transfer

The ability to hold state information is a basic requirement of a general purpose computational device [Menabrea, 1842], [Babbage, 1864]. Such working memory allows the storage of intermediate information, thus allowing a computation to be performed as a series of steps over time, which consult and manipulate this memory. The simplest form of state can be encoded in a flip-flop device which holds a representation of either a `0` or a `1`. Although such a binary switch has a modest capacity, by connecting a number of them together one can attain a state space of exponential size with respect to the number of binary switches used. Nature, too, makes use of binary switches, such as the bistable genetic switch that is used by the well-studied bacteriophage λ [Ptashne, 2004]. The switch decides whether a virus should become *lytic* – duplicate and break open the cell membrane releasing copies of itself, or *lysogenic* – integrate itself into the DNA of the bacterium [Herskowitz and Hagen, 1980], [Ptashne *et al.*, 1980]. The switch is implemented as a mutually inhibitory network: two proteins, Cro and CI, the relative concentrations of these representing a state, suppress each others' production. Two of the first synthetic biological devices to be produced were also bistable switches, which operated in bacteria [Gardener *et al.*, 2000]. An illustration of one of these switches, which used mutual repression, is shown in Figure 2.2a.

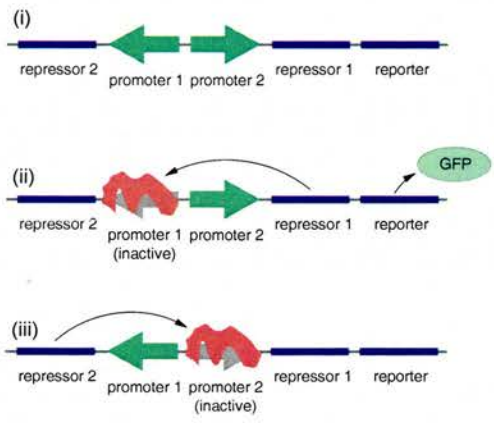


Figure 2.2a: An illustration of a genetic toggle switch constructed by Gardener *et al.*. (i) Two promoters, shown in green, each point towards a protein encoding, shown in blue, which can suppress the other's production. (ii) When *promoter 2* is active, the *repressor 1* protein is produced which deactivates *promoter 1*, thus suppressing the production of *repressor 2*. Green fluorescent protein (GFP) is produced to report this stable state of the switch. (iii) The alternative stable state occurs when *promoter 1* is active and is producing *repressor 2* which deactivates *promoter 2*, thus suppressing the production of *repressor 1*. No GFP is produced when the switch is in this state.

Unfortunately, it does not appear to be as straightforward to create a larger state space by connecting a number of these bacterial switches together. Unlike their electronic analogues, biological regulatory signals – proteins – will diffuse from one part of a cell

to another. It may therefore be necessary to represent each signal with a different protein, and in doing so, increase the complexity of the synthetic device. Alternatively, it may be possible to utilise the organised compartmentalisation employed by eukaryotic cells to prevent cross-talk between devices with similar protein signals. Recently, a synthetic switch has been incorporated into eukaryotic cells [Kramer, 2004] supporting the validity of this approach. A second approach towards compounding the state space that can be represented by switches is to distribute state over a collection of cells. A consequent problem is the need for a mechanism to communicate this distributed state information between cells.

The quorum sensing bacteria, *Vibrio fisheri*, possess a well-studied intercell signalling mechanism [Bassler, 1999]. Each bacterium secretes an *autoinducer*, a signalling molecule, which can be detected by nearby cells [Fuqua *et al.*, 1994]. When a colony of bacteria is sufficiently dense, the concentration of the autoinducer rises above a threshold which activates regulatory machinery causing the cells to fluoresce. A synthetic intercell communication system has also been developed [Weiss *et al.*, 2003] based on *Vibrio fisheri*. LuxI autoinducer molecules were secreted by a population of cells induced by aTc molecules. These small autoinducer molecules were received by a separate population of cells and this activated regulatory machinery causing the production of a fluorescent protein (Figure 2.2b).

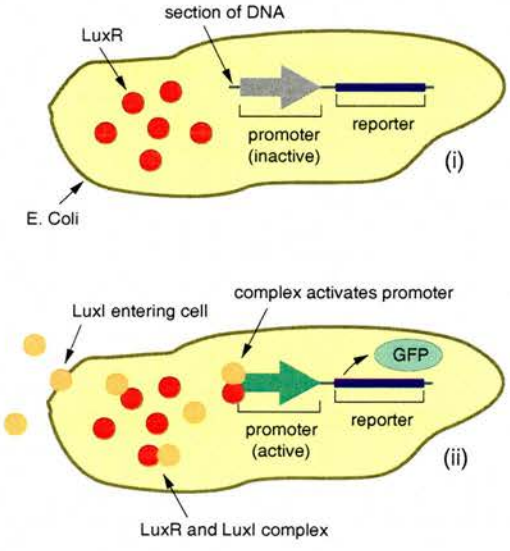


Figure 2.2b: An illustration of a receiver cell demonstrated by Weiss and Knight in an intercell communication mechanism. (i) This receiving cell produces the LuxR protein and, in the absence of an external signal from a transmitting cell, a promoter which points towards a reporter protein is inactive. (ii) LuxI proteins, which are produced by transmitting cells, are able to pass through the receiving cells' membranes and form complexes with LuxR proteins. These complexes can bind to the promoter and cause its activation, resulting in the production of the GFP reporter.

However, such intercell communication also brings signal diffusion problems, but between cells, rather than within a cell. One useful discrimination is to be able to de-

termine whether a signal came from a nearby cell, thereby increasing the ability to separate signals by location. Recently, a synthetic pulse-generating network has been constructed which allows cells to detect transmissions from nearby cells while ignoring communication from cells farther away [Basu *et al.*, 2004]. This system exploits the fact that that the rate of a received, diffused signal's concentration change is higher, the closer the transmitter is. The mechanism of the receiving cell, shown in Figure 2.2b, relies on a race condition between expression of a fast-decaying fluorescent output signal, and the build up of a fast-decaying protein which suppresses expression of this output signal.

When considering state over generations of cells, rather than individual ones, nature employs DNA to encode developmental and behavioural information. Unlike protein signals, which encode information which diffuses around a bacterial cell and which have no fixed location, the quaternary information encoded in DNA is fixed in order by the DNA backbone, thus creating a string of data, where location can be exploited: even though there are only four kinds of nucleotide base in a DNA sequence, by connecting a number of nucleotide bases together, one gains a state space of exponential size with respect to the length of the sequence.

2.3 Time

A second important component of a computational device is a clock, to enable a series of instructions to be carried out in sequence and to allow the synchronisation of events. One particularly notable example, found in nature, is the cell cycle, which controls the growth and division of cells [Murray and Hunt, 1993]. This process is partitioned into a number of distinct phases, in which the transition to the next phase is regulated by checkpoints which assess factors such as whether DNA damage has occurred or whether environmental conditions favour duplication. Another example is that of circadian rhythms – daily cycles of physiological activity [Winfree, 2000] that can be synchronised with environmental cues, e.g. exposure to sunlight. One of the first synthetic biological devices created was an oscillator [Elowitz and Leibler, 2000]. A cycle of three genes was made to form a negative feedback loop (Figure 2.3a): *lacI* inhibiting *tetR*, *tetR* inhibiting *cI* and *cI* inhibiting *lacI*. For suitable choices of parameters such as protein decay rates, an unstable state of temporal oscillations in the three protein concentrations was demonstrated.

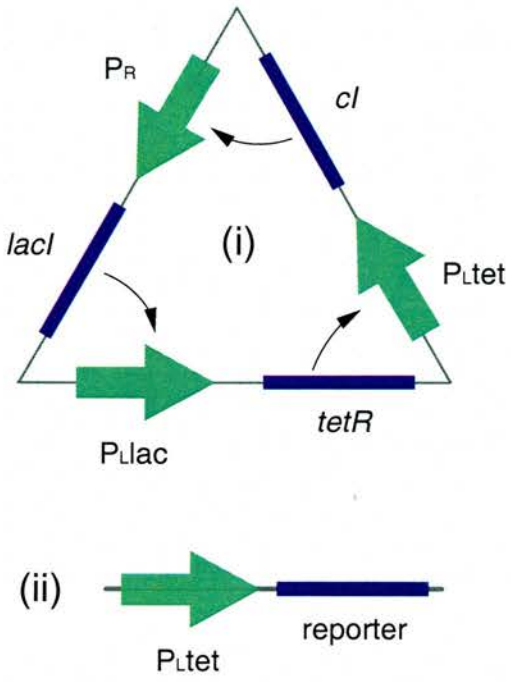


Figure 2.3a: An illustration of the design of the ring oscillator demonstrated by Elowitz and Leibler. (i) A cycle of three, fast-decaying proteins which suppress each others' production. For example, when the $P_L$lac promoter is active, the protein TetR is produced which suppresses the $P_L$tet promoter. When the $P_L$tet promoter is suppressed, no CI will be produced, therefore the P_R promoter will be active which will allow the production of LacI. LacI will now cause suppression of the $P_L$lac promoter which began this phase of the cycle. Thus, the activities of the promoters oscillate and consequently so do the repressor protein concentrations. (ii) One of the repressor proteins, TetR, was made to suppress the production of a reporter protein (GFP) to aid the monitoring of state of the ring oscillator.

An important feature of *in silice* clocks is their sharp, unambiguous edges signifying a clear transition from one state to the next. One notable feature of the artificial clock, and many other synthetic devices, is their tendency to exhibit noisy signals, possibly due to stochastic fluctuations of protein concentrations and reactions. However, natural cell mechanisms do show that they are able to cope remarkably well in the face of constantly fluctuating environmental conditions, perhaps owing to the extensive checks and balances they employ. For example, a protein is *self-regulating* if its production is affected by a negative feedback loop [Ptashne *et al.*, 1976], [Becskei and Serrano, 2000], i.e., the higher the concentration of a protein is, the less of it will be produced (Figure 2.3b).

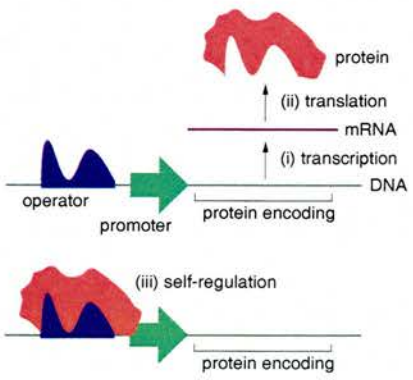


Figure 2.3b: An illustration of the self-regulation of a protein. (i) A RNA polymerase molecule (magenta) creates mRNA transcripts by reading DNA from a promoter (start-site, green). (ii) Ribosomes translate the transcript producing proteins. (iii) In this example, there is a site which the protein is able to bind to next to the promoter (an operator). When bound, the protein inhibits the attachment of RNA polymerase molecules to the nearby promoter, thus negatively regulating its own, further production.

2.4 Modularity

Nature has endowed cells with innumerable capabilities that enable them to sense and interact with their environment. Bacteria use chemotaxis to swim towards the source of an attractant such as glucose [Macnab and Koshland *et al.*, 1972], [Segall *et al.*, 1986]: they change their direction by tumbling in a random orientation until they point towards the highest concentration gradient of the attractant [Berg, 2004], [Amos *et al.*, 2006]. Conversely, cells can use chemotaxis to swim away from a repellent that is poisonous. Magnetotactic bacteria can also sense and migrate along the magnetic field of the Earth towards favourable habitats [Blakemore, 1975]. When the environment does not favour the survival of a colony, its bacteria can form endospores [Tyndall, 1877], which provide protection against a range of harsh conditions that include: a lack of water, starvation and exposure to ultraviolet light [Nicholson *et al.*, 2000]. Bacteria can sense temperature changes and adjust their physiological processes accordingly to survive and grow [Eriksson, 2002].

With the advent of recombinant DNA technology it is becoming possible to incorporate natural devices, which have been sufficiently well characterised, from one type of organism into another. For example, the genes which cause fireflies to bioluminesce can be inserted into genetic networks to report an aspect of their regulatory state [Gould and Subramani, 1988]. It is also possible to adjust or optimise natural devices for a specific purpose. For example, *E. coli* cells which have developed a natural and accurate detection mechanism for the toxin arsenic have been genetically modified to fluoresce to signify its presence, thus resulting in a human-readable detection kit for water purity [Stocker *et al.*, 2003]. A quite different sensor has been synthesised in yeast cells [Shimizu-Sato *et al.*, 2002] which can be activated and deactivated by short pulses of light at two different wavelengths, with switching response times of under one second (Figure 2.4). The sensor's state of activation can in turn affect the production of genes inside a cell.

From an engineering perspective, one of the most exciting aspects of biological devices is their potential in the long term to be used as modular components which can be arranged and connected in accordance with distinct input and output signal interfaces, in an arbitrary design [Guet *et al.*, 2002], [Dueber *et al.*, 2003].

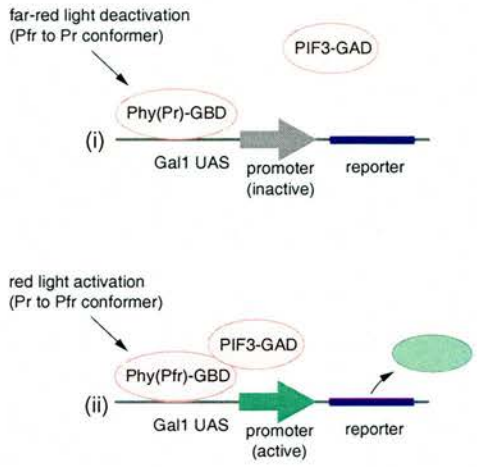


Figure 2.4: An illustration of the light-responsive promoter system made by Shimizu-Sato *et al.* in 2002. (i) A section of DNA contains an inactive promoter which points towards the encoding of a reporter protein. Phy(Pr)-GBD fusion protein is synthesised by the cell, a chromophore is attached and this attaches to the DNA at the promoter site (Gal1 UAS). The cell also synthesises the protein PIF3-GAD which, in this upper diagram, is unable to react with other molecules. (ii) When the cell is exposed to red light, phy(Pr)-GBD is photoactivated, within one second, creating phy(Pfr)-GBD which has a strong affinity for PIF3-GAD. When PIF3-GAD attaches to phy(Pfr)-GBD, this activates the promoter, turning on the reporter. Phy(Pfr)-GBD can also be deactivated through exposure to far-red light to return to its original Pr conformity shown in the upper diagram.

Of course, there are significant hurdles to be overcome before this can be realised generally, not least being the ability to predict the behaviour and stability of a combination of designed devices residing inside a host cell. However, the range of synthetic biological components is steadily increasing, as is the categorisation of their engineered component parts: the on-line BioBricks repository catalogues a growing number of standardised and interchangeable biological parts ['BioBricks'] and aims to make the process of assembling genetic components more reliable and predictable [Knight, 2001].

2.5 Persistence

The ability to encode programs and other data on permanent media, such as Joseph-Marie Jacquard's punched cards, which controlled the first automatic loom able to weave complex patterns in cloth, increases the autonomy and flexibility of a computational device by reducing the need for manual interaction, when providing the device with input [Campbell-Kelly, 2004]. Conventional electronic computers typically have a stored program architecture [Neumann, 1945], which encodes programs in the same manner as data [Turing, 1936]. This allows the computer to be used as a general purpose

device as it can be reprogrammed by supplying it with data that encodes a program. Nature's own brand of permanent media is DNA, encoding the developmental and behavioural programs of cells. One can consider DNA to be the persistent storage device of a cell that passes information from one generation to the next [Shapiro, 2002]. Within the life-time of a cell, nature has evolved several methods of modifying and rearranging DNA, in such a way that new information is encoded, which may consequently affect the operation of the cell.

2.5.1 Restriction Endonucleases

Bacteria have developed elaborate defence mechanisms to protect themselves from viral infection. For instance, bacteria can synthesise enzymes called *restriction endonucleases* which recognise a part of the encoding sequence of an invading DNA virus [Luria, 1970], and digest its DNA, near the place of recognition [Smith and Wilcox, 1970], [Kelly and Smith, 1970]. When a restriction endonuclease digests DNA and cleaves both strands, *cohesive ends* (Section 1.5.4) are produced. Cohesive ends may be rejoined with a complementary sequence (Section 1.5.2), having the same type of overhang on the upper or lower DNA strand, with the assistance of *DNA ligase* (Section 1.5.4). In this way, it is possible to digest some DNA, to break it apart at a specific place, then to insert another piece of DNA, with compatible ends, in between the breakage and ligate the two pieces of DNA together [Lobban and Sutton, 1973]. This *modular* method of DNA manipulation has been harnessed *in vitro* and is used extensively during standard cloning procedures [Roberts, 2005], [Sambrook and Russel *et al.*, 2001].

Recently, an *in vitro* molecular computer [Adleman, 1996] that encodes its program, input, transient states and outputs all using nucleic acids, and makes use of a restriction endonuclease, *Fok*I, isolated from *Flavobacterium okeanokoites*, to effect autonomous state transitions, has been demonstrated [Benenson *et al.*, 2004]. The automaton detects mRNA disease indicators, the presence of which causes it to progress through a series of states, until either all indicators are detected (a positive diagnosis), or otherwise (a negative diagnosis). Indicators are encoded, in series, in DNA referred to as the *diagnosis* DNA. When an indicator is detected, a transitional fragment of DNA is able to bind to the diagnosis DNA. These transitional fragments contain a specific sequence that is recognised by *Fok*I, which digests the diagnosis DNA a fixed distance from this sequence. In doing so, the next transitional fragment is released thus causing a state

transition. The final fragment of the diagnosis DNA contains a *hairpin loop* – a single strand of DNA that has self-complementary ends. The base pairs of these ends can bond with each other to form a section of dsDNA, and in doing so, this causes the unpaired ssDNA to form a loop. If all indicators are detected, then the last digestion performed by *Fok*I releases the ssDNA, which is able to function as a drug. In isolation, this mechanism would be subject to false-positive diagnosis owing to the stochastic nature of the detection process. Consequently, a ssDNA drug suppressor is released in the event of a negative diagnosis. Furthermore, this mechanism may be tuned to alter the threshold of detection by adjusting the ratio of the ssDNA drug and its suppressor released when a corresponding positive or negative diagnosis is made by an automaton, out of a number of automata running the same diagnosis program. The molecular mechanism used did not require the use of DNA ligase to rejoin severed DNA, which consumes energy; rather, it was discovered that *Fok*I, at least *in vitro*, is able to digest DNA over a *hybridised* join (Figure 2.5.1), where complementary cohesive ends are aligned and weakly bonded, but where covalent bonds have not been formed by DNA ligase [Benenson *et al.*, 2003].

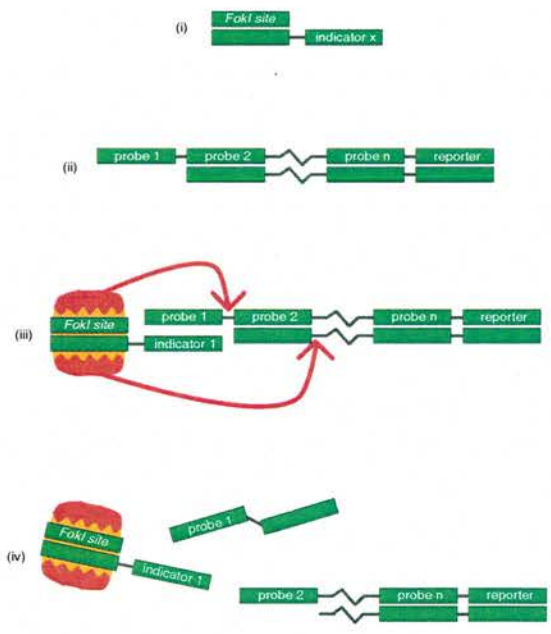


Figure 2.5.1: An illustration of the molecular automaton demonstrated by Benenson *et al.* which uses the restriction endonuclease, *Fok*I, over annealed cohesive ends. (i) The format of a state transition molecule which contains a dsDNA *Fok*I recognition site upstream of a cohesive end with a specific binding pattern (indicator). (ii) The format of the input DNA which contains a series of sequences (probes) which may match a state transition molecule. (iii) *Fok*I attaches to a state transition molecule which, given compatible ends, also anneals to the input DNA. Despite the absence of DNA ligase, *Fok*I digests DNA a fixed distance away from its attached site over the non-covalent bonding. Consequently, the input molecule is truncated and the next *probe* in the series is exposed, ready for the next possible transition. If all state transitions are completed, then a final sequence is released which corresponds to this system's output (reporter).

The nucleotide base sequence patterns that restriction endonucleases recognise are often *symmetrical*, in that they read the same on either complementary strand of DNA. Some patterns are ambiguous (Section 1.5.6) and the majority of recognition sequence patterns contain between four to eight nucleotide bases. For example, there are 64 potential symmetrical patterns which can be composed using six nucleotide bases ($4^{(\frac{6}{2})}$). So far, enzymes recognising 56 of these have been discovered [Roberts *et al.*, 2003]. *Neoschizomeric* restriction endonucleases, which recognise the same pattern, but digest DNA at different places, are also common.

A random DNA sequence of six bases, where each base can be one of four possible nucleotides, will match an unambiguous pattern of the same size with a probability of $\frac{1}{4096}$ (4^{-6}). Many bacterial genomes are composed of more than a million nucleotide bases [Fleischmann *et al.*, 1995], [Kunst *et al.*, 1997]. For example, the bacterial strain, *E. coli K-12*, which is used routinely in the laboratory, holds over four million bases [Blattner *et al.*, 1997]. A chosen six nucleotide base pattern will fail to match a random, four million base pair sequence with a probability of 6.8×10^{-425} ($(1 - \frac{1}{4096})^{4\times10^6}$). This provides an *indication*, that in the absence of protective measures, a bacterium's DNA is very likely to be digested by an arbitrary restriction endonuclease which recognises a typically-sized, six base nucleotide pattern. One method of protection, available to both bacteria and viruses, is the mutation of nucleotide bases within the region of a recognition pattern [Luria and Delbruck, 1943]. Given the redundancy of the mapping from *codons* – triples of nucleotide bases – to amino acids which compose proteins [Crick, 1958], it is sometimes possible to change a base without altering the amino acid produced, or otherwise to substitute an encoding of an amino acid with adequate properties for the operation of the target protein. Similarly, it is possible to mutate other parts of the DNA which do not encode amino acids, provided this does not adversely affect a vital regulatory mechanism. In bacterial cells, the replication of circular DNA typically begins at a single location, called an *origin of replication*, and proceeds in both directions, on each strand, away from this location [Messer, 2002]. Therefore, when the DNA of a bacterium is broken, this prevents its complete replication. Given this, if a restriction endonuclease is synthesised by a bacterium, selective pressure will favour DNA mutations which prevent the digestion of its own DNA.

2.5.2 Methylases

Methylases are enzymes that can recognise nucleotide base sequence patterns, in the same manner as restriction endonucleases [Bird, 1992]. They add methyl groups to recognised patterns which prevent the subsequent digestion of DNA at these locations [Nelson and McClelland, 1989]. Bacteria methylate their DNA after they have reproduced and it is relatively unlikely that viral DNA will be methylated within this period. This *restriction-modification* system helps bacteria disrupt viral invasion without damaging themselves (Figure 2.5.2).

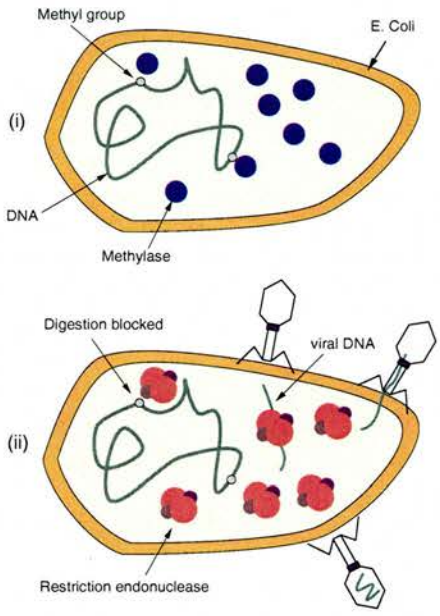


Figure 2.5.2: An illustration of how methylation assists bacteria in fending off viral invasion. (i) Soon after a bacterium has reproduced, it releases methylase enzymes which recognise specific base sequences and protect them from digestion by adding methyl groups. (ii) The methylase proteins decay and the bacterium synthesises restriction endonucleases, which are unable to digest the bacterial DNA at the sites that have been methylated. However, when viral DNA is injected into the bacterium, the restriction endonucleases are able to digest the invading DNA at sites which they recognise.

2.5.3 Homing Endonucleases

In contrast to the low specificity exhibited by restriction endonucleases, another class of enzyme, *homing endonucleases*, are able to digest DNA inside recognition patterns of between 12 to 40 nucleotide bases [Belfort and Roberts, 1997]. These longer patterns are considerably more specific and will match a random nucleotide sequence of equivalent size with an approximate probability of between one in ten million ($\approx 4^{-12}$) and one in a million, billion, billion ($\approx 4^{-40}$). However, in some instances, a nucleotide base mismatch may reduce the affinity of a homing endonuclease for its pattern, rather than completely preventing recognition [Seligman *et al.*, 2002]. The genes of these rare-cutting enzymes are interesting, parasitic genetic components which cause the digestion of copies of DNA which do not contain them [Burt and Koufopanou, 2004]. In do-

ing so, a cell's DNA repair mechanism is triggered and a homing endonuclease gene (HEG), which is stored on the undigested DNA, is copied into the place of the broken strand. Once repaired, the manipulated DNA copy is no longer susceptible to digestion, as the HEG interrupts *its own* recognition sequence (Figure 2.5.3).

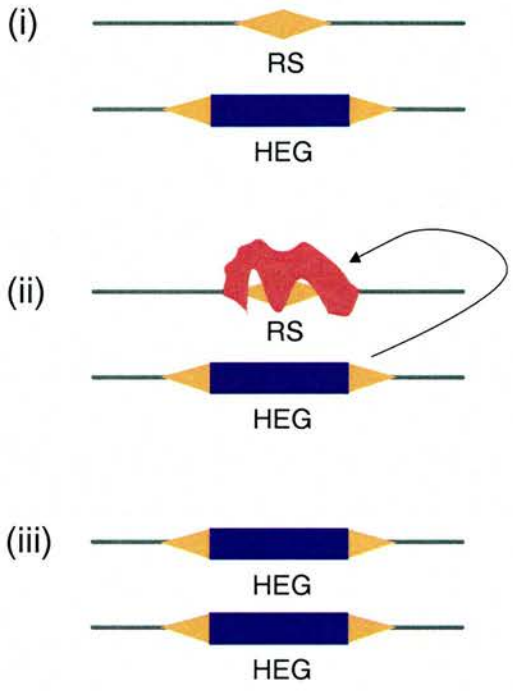


Figure 2.5.3: An illustration of how homing endonuclease genes propagate across chromosomes, closely based on Burt and Koufopanou's diagram. (i) Two homologous sequences, one without a HEG encoding but with a recognition site (upper) and one with a HEG encoding interrupting a recognition site (lower). (ii) Synthesis of the HEG creates a protein which causes digestion of the upper strand at the recognition sequence. The lower strand resists digestion as its recognition sequence is interrupted. (iii) The recombinatorial repair system uses the uncleaved DNA to be used as a template to repair the cleaved DNA and the HEG is consequently copied across. Now, neither DNA sequence can be digested by this HEG.

This "selfish" strategy has been employed by homing endonuclease genes in a variety of situations: across chromosomal DNA copies; in bacterial viruses, when two viruses attack the same cell and in eubacteria, although it is not yet known how this strategy succeeds in these organisms. Some homing endonucleases are also currently being used as genetic engineering tools [Jasin, 1996], although, so far, only 50 of these enzymes have had their recognition pattern and precise cleavage locations determined [Roberts *et al.*, 2003], which is considerably fewer than those in the case of restriction endonucleases.

2.5.4 Recombinases

Yet another class of DNA-arranging enzymes, which is used extensively in current research, is the recombinase family [Stark *et al.*, 1992]. These enzymes recognise pairs of highly specific DNA sites and can excise, insert, invert and translocate DNA, depending on the orientation and location of the recognition sites (Figure 2.5.4). For example, the *integrase*, Int, is used by bacteriophage λ to integrate its DNA with a host bac-

terium when it lies dormant in its lysogenous mode, and the *excisionase*, Xis, is used to excise its DNA during lysis [Weisberg and Landy, 1983], [Sadowski, 1986].

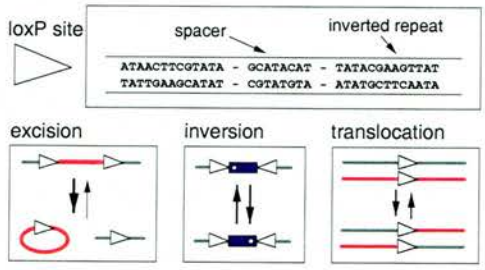


Figure 2.5.4: An illustration of the DNA-arranging properties of the recombinase, Cre, and its loxP recognition sites. (upper) The sequence of the loxP site consists of two inverted, 13 nucleotide base sequences which are separated by an asymmetric eight nucleotide base sequence. (lower) Depending on the orientations and locations of two loxP sites, Cre can rearrange DNA by excision (and its inverse reaction, insertion), inversion and translocation.

The recombination mechanism of the Cre enzyme and its loxP recognition sites, from bacteriophage P1, has been applied to mouse cells [Lakso *et al.*, 1992], [Orban *et al.*, 1992] and has become a useful research tool allowing the modification of DNA *in vivo*. Transgenic mice can be engineered to express Cre in specific tissues of interest, or at a particular stage of their development. A second line of mice can be engineered such that two, similarly oriented, 34 nucleotide, loxP sites flank a section of DNA. In the offspring resulting from breeding these two lines of mice together, when cells express Cre, the section of DNA flanked by loxP sites is excised. The section of DNA is typically a gene, the regulatory effect of which is investigated by its targeted *knock out*. Conversely, DNA can also be rearranged to place a gene downstream of a promoter. Although Cre-mediated inversion of DNA between loxP sites is reversible, mutants of these recognition sites have been created which strongly favour one orientation over another [Oberdoerffer *et al.*, 2003].

Arguably, the most remarkable and extensive DNA rearrangement mechanism is employed by the immune systems of vertebrates [Janeway, 2001], [Schatz, 1999]. In order to detect bacteria, viruses and toxins, cells of the immune system are provided with sites that bind to antigens – material that is suspected to be harmful to the body. Segments of *immunoglobulin* genes, which are responsible for the particular specificities of antigen-binding sites, can be randomly transposed by the enzymes RAG1 and RAG2 to form a potential of around one hundred million specificities [Pollard and Earnshaw, 2004 *(pages 480-482)*]. To address the problem that the immune system needs to be able to respond to a wide range of antigens, which it may not have encountered previously, a brute-force approach towards traversing the large potential antigen-binding

search space is taken, by producing cells with a wide variety of immunoglobulin genes. When antigens are detected, the immune system proliferates clones of the cells which successfully recognised them, and in doing so, specialises its defensive response.

In contrast to restriction endonucleases, only a small number of recombinases have been adopted for routine human uses, principally Cre, as already described in Section 2.5.4, and Flp from the yeast, *Saccharomyces cerevisiae* [Sadowski, 2003]. Also, the ones discovered so far lack the *modularity* of cohesive ends, where compatible ends can be created by cleaving two different sequences. However, recombinases have been proven to work effectively and efficiently across a range of engineered cell types, including mammalian cells [Kilby *et al.*, 1993]. Both endonucleases and recombinases with artificially modified nucleotide specificities are being developed, using *evolutionary protein* techniques [Collins *et al.*, 2003]. The genes of DNA-arranging enzymes are mutated or hybridised and experiments are composed to select for those modified enzymes with altered recognition site specificities. Consequently, the repertoire of DNA-arranging enzymes is expanding, not only through their discovery in natural systems, but also through efforts to create synthetic proteins with new recognition patterns.

2.6 Designing Synthetic Devices

2.6.1 Overview

Ideally, when one is designing a *de novo* device, whether electronic or biological, a repertoire of well-defined building blocks should be available, together with representations which can predict the behaviour of devices composed of these building blocks [Simpson *et al.*, 2004]. In biology today, the fabrication of *de novo* devices is often a time consuming and costly process. For example, it is reported that Gardener *et al.*'s synthetic toggle switch took almost one year to build [Gibbs, 2004]. Predictive models of cellular behaviour will aid our understanding of natural systems [Jong, 2002] and have the potential to direct the design process of *de novo* devices in promising directions. It has recently been argued that typical cellular simulation models have been based on incomplete or simplified biological knowledge, but that advances in computational modelling and biological understanding will pave the way towards useful and predictive models [Endy and Brent, 2001]. For instance, the application of model checking techniques based on incomplete biological data [Clarke *et al.*, 1999] allows models

to be more readily tested and enhanced [Koch *et al.*, 2005], [Batt *et al.*, 2005]. Similarly, biological databases are being constructed which describe cellular reactions at the level of proteins and DNA sites [Salgado *et al.*, 2004], [Keseler *et al.*, 2005].

2.6.2 Modelling Approaches

A range of approaches towards the modelling of *in vivo* reactions has been explored including discrete, binary networks [Jacob and Monod, 1961], [Kauffman, 1974], piecewise linear [Tchuraev, 1991], non-linear [Yagil and Yagil, 1971] and physical-chemical [Smith *et al.*, 1977]. Recent work also illustrates the benefits of adopting an object-oriented approach towards biological modelling including: abstraction, inheritance hierarchies and object reuse [Webb and White, 2005], [Meir *et al.*, 2002]. Of particular interest, typical *in vivo* molecular reactions occur by random collisions, often with small number of each species of molecule present [Kramers, 1940]. Consequently, stochastic modelling approaches [Turner *et al.*, 2004] are being adopted [Arkin and Ross, 1994], [McAdams and Arkin, 1997] to investigate, and account for, the consequences of cellular signals being represented by small, discrete numbers of particles, [Rao *et al.*, 2002]. The stochastic simulation algorithm (SSA) is an efficient procedure that allows the numerical simulation of a well-mixed population of reactants over time while accounting for the randomness of biochemical reactions [Gillespie, 1977]. Increased modelling accuracy, beyond the SSA, has been obtained by the inclusion of the individual states of molecules and a spatial structure, allowing nearest-neighbour interactions [Le Novère and Shimizu, 2001], [Lipkow *et al.*, 2005]. In general, as the accuracy and scale of a model increases, so do both the computational demands and the need to supply more detailed information about molecular behaviour. However, at present, when modelling *de novo* devices, synthetic biologists have typically relied on simple mathematical models of transcription as not enough is known about cellular molecular interactions to compose more precise, and testable, representations [Elowitz and Leibler, 2000]. The modelling approach adopted in this thesis addresses the requirements for the efficient simulation of the dynamic rearrangement of DNA programs, and for mapping between a symbolic representation and network of interacting DNA sites and proteins. Justification for the implementation of a bespoke simulation environment is presented in Section 3.2.

2.6.3 Regulatory Noise

Nature has produced a diverse repertoire of cellular biological devices. The signals and memory of cellular devices are represented by small numbers of molecules, typically between one and one thousand, per signal, per cell. The interactions of these signals with a cell's regulatory mechanisms are stochastic in character. Consequently, at the level of individual genes, gene regulation is inherently noisy. Noise is sometimes desirable: for example, noise during the process of DNA replication can introduce mutations, which aid evolution, and noise may also contribute towards phenotypic diversity [Spudich and Koshland, 1976], [Levin, 2003]. However, from a traditional engineering perspective, noise can erode information and introduce error. In order to create a library of well-defined components for the design of *de novo* devices, it is therefore important to characterise the regulatory noise that affects the behaviours of components. Such characterisation not only helps us to understand the operational limits of components but also is a step towards the eventual aim of reducing the effect of noise, either through the modification of components, or by assembling noise-tolerant networks of components. For example, synthetic self-regulating networks have been shown to increase the stability of gene expression by using negative feedback [Becskei and Serrano, 2000], and a proposed cell-to-cell synchronisation mechanism would allow a population of cells to couple their oscillatory devices [McMillen *et al.*, 2002].

To address the question of how stochastic gene expression (intrinsic stochasticity) and fluctuations in other cellular components (extrinsic stochasticity) contribute towards noise, Elowitz *et al.* designed an experiment able to distinguish between these two factors in *E. coli* [Elowitz *et al.*, 2002]. Two reporter genes encoding, cyan (CFP) and yellow (YFP) fluorescent proteins were inserted equidistant from an *E. coli* origin of replication and placed under the control of two identical lac operons. The amount of expression of the fluorescent reporters could be adjusted by adding different amounts of IPTG. Intrinsic noise was measured as the degree of correlation between the two reporter proteins in a single cell. It was found that this intrinsic noise was inversely proportional to the level of gene expression. Extrinsic noise, measured as the fluctuation of signals over time, in comparison, peaked at intermediate levels of gene expression and then declined as the level of gene expression increased. The authors suggested that the cell-to-cell variation in the concentration of LacI, which regulates the lac promoter and is activated by IPTG, as a possible explanation for the peak in extrinsic noise.

A related question, addressed experimentally by Ozbudak *et al.* (2002), is what the comparative effect of translational and transcriptional efficiency is on the variability (noise) of bacterial gene expression [Ozbudak *et al.*, 2002]. A set of sequences with different translational efficiencies was created with point mutations in the initiation codon and ribosome binding site of a reporter gene, *gfp*. Similarly, a set of transcriptional efficiencies was produced with point mutations made to the promoter upstream of the reporter gene. The authors found that increased translational efficiency is the predominant source of noise. This result complements the findings of an earlier stochastic model for the origins of noise in prokaryotic cells [McAdams and Arkin, 1997]. In contrast to this finding, noise arising from transcription in eukaryotic cells can contribute significantly to the heterogeneity of a clonal population, and this noise can be regulated by translation [Blake *et al.*, 2003]. Of particular interest to those seeking to engineer *de novo* synthetic devices, Ozbudak *et al.* (2002) suggest that several genes, which are inefficiently translated and consequently waste energy, have been naturally selected for their lower noise characteristics, such as *cI* in bacteriophage λ .

2.6.4 Evolutionary Processes

In contrast to the rational design of synthetic networks, a combinatorial approach towards the construction of libraries of genetic networks has been demonstrated [Guet *et al.*, 2002]. Networks were composed of three regulatory protein encodings, each coupled with one of five different promoters, which were affected by these proteins (Figure 2.6.4a).

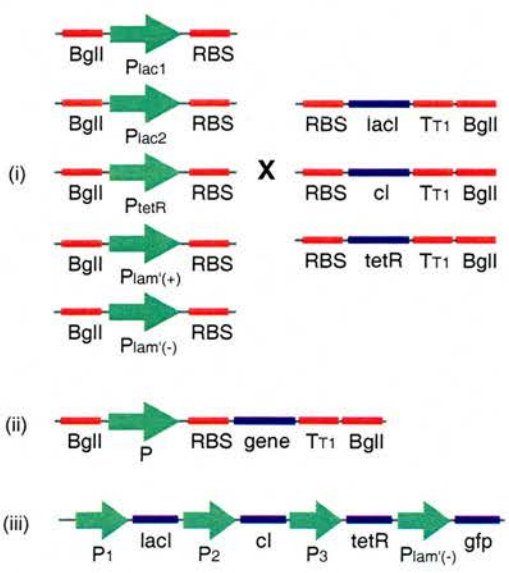


Figure 2.6.4a: An illustration of a combinatorial approach used to synthesise gene networks. (i) A set of five promoters and three genes are flanked by a *BglI* restriction site and a RBS. (ii) By performing fusion PCR, utilising the identical RBS as internal primers, 15 possible promoter-gene pairs were created of this form. (iii) The *BglI* restriction sites were designed such that, after digestion, different cohesive ends were produced. Next, when ligating digested pairs together, an order was enforced on the gene arrangements (*lacI-cI-tetR*) through the compatibilities of the cohesive ends. The triples of gene-promoter pairs were inserted into a vector containing a *gfp* gene which could be down regulated by λ CI.

Fusion PCR [Vallejo *et al.*, 1994] was used to create all 15 possible promoter-gene pairs. Sites recognised by the restriction endonuclease, *BglI*, were incorporated into PCR primers so that digested amplified pairs would have specific cohesive ends ensuring that the order of the three regulatory genes was fixed when pairs were ligated. Isolated cultures holding plasmids with three promoter-gene pairs were analysed by subjection to combinations of the presence and absence of two input signals: IPTG and aTc. Each plasmid held a *gfp* gene regulated by the repressible *cI* promoter of bacteriophage λ and output measurements were obtained by detection of GFP. Of the 125 (5^3) possible network combinations that could potentially be created, 30 distinct combinations were isolated, representing 13 distinct network connectivities. The authors also noted that some networks with different connectivity exhibited qualitatively similar behaviour and concluded that the behaviour of simple genetic networks composed of a small number of well-characterised components cannot always be inferred from network connectivities.

A hybrid approach mixing the rational design of biological devices with rounds of targeted random mutations, followed by screening for behavioural characteristics, has also been demonstrated [Yokobayashi *et al.*, 2002]. This evolutionary approach was applied to the optimisation of a synthetic genetic *inverter* – a logic gate that produces a binary output that is the opposite of its binary input [Weiss and Basu, 2002] (Figure 2.6.4b).

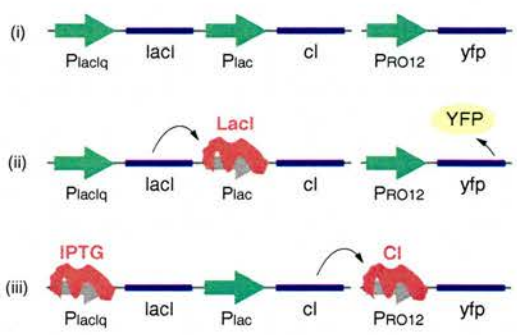
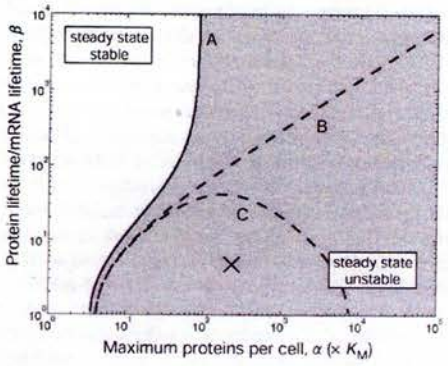


Figure 2.6.4b: The gene regulatory layout of an inverter. (i) On one plasmid, a *lacI* gene is controlled by a P_{laclq} promoter and a *cI* gene is controlled by a P_{lac} promoter. On a second plasmid, a reporter gene, *yfp* is regulated by a P_{RO12} promoter. (ii) In the absence of IPTG, LacI is synthesised which causes the suppression of the P_{lac} promoter, inhibiting CI production. In the absence of CI, P_{RO12} is active and YFP is produced. (iii) When IPTG is present, it downregulates the production of LacI. In the absence of LacI, CI is synthesised, which inhibits production of YFP by downregulating the P_{RO12} promoter.

For the inverter to operate correctly, the concentration of CI must be high enough to repress the λ P_{RO12} promoter in the presence of a given amount of IPTG, which indirectly causes the production of CI. The converse case must also be true: in the absence of IPTG, the amount of CI produced must be low enough that the λ P_{RO12} promoter is not repressed. Fine-tuning biochemical parameters of even simple functional device interfaces is not a trivial problem owing to the complex molecular interactions which occur inside cells. Consequently, it was demonstrated how an initially untuned, and non-functional, inverter could be rapidly turned into a functional device by targeted mutations of the *cI* gene and its RBS. Colonies of *E. coli* containing the mutated inverters were grown on a plate with IPTG, and those which fluoresced were transferred to a second plate without IPTG. Colonies which did not fluoresce on the second plate were isolated as candidates containing functional inverters. On sequencing, it was found that a frequent successful modification was the addition of a stop codon inside the *cI* gene, in its C-terminal domain, which is responsible for CI protein *dimerisation* – the pairing of CI proteins. Dimerised CI proteins attach to the λ P_{RO12} promoter site and cause its repression; thus, it is possible that by disrupting the cooperative binding mechanism, the effectiveness of CI repression was reduced.

2.6.5 Synthetic Network Models

The possible classes of behaviour of various synthetic gene networks have been predicted and explained using simple mathematical models [Elowitz and Leibler, 2000], [Gardener *et al.*, 2000], [Becskei and Serrano, 2000]. These models have indicated experimental parameters which may require adjustment to ensure the intended operation of a *de novo* network. For example, Elowitz and Leibler's synthetic oscillatory network was characterised using six coupled differential equations (Figure 2.6.5a, *right*). Repressor protein and mRNA concentrations are represented as continuous dynamical variables, p_i and m_i respectively. α_0 represents the amount of protein synthesised from a fully-repressed promoter, and α represents the additional amount of protein synthesised from an unrepressed promoter. n corresponds to a Hill coefficient and β is the ratio of protein and mRNA decay rates. A state of the network is considered to be *steady* when the rate of each repressor protein's production equals its rate of degradation. A state is considered to be *stable* when the system remains in that state, in the absence of any externally introduced signals.



Reproduced with permission from [Elowitz and Leibler, 2000] © Nature Publishing Group (www.nature.com).

Figure 2.6.5a: The division of steady regions and stable regions of the ring oscillator when varying n, the Hill coefficient indicating the co-operativity of repression, and α_0, the expression resulting from a fully-repressed promoter (*left*). The region boundaries are defined as: A, n=2.1, $\alpha_0 = 0$; B, n=2, $\alpha_0 = 0$; C, n=2, $\alpha_0 = 10^{-3}$.

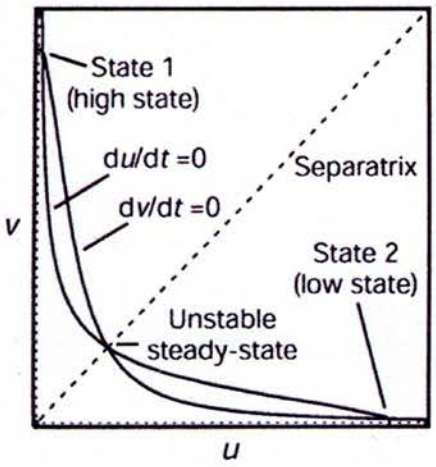
The template for the the pairs of differential equations used to model the synthetic oscillator (*right*) takes the following instantiations: i=*lacI*,j=*cI*; i=*tetR*,j=*lacI*; i=*cI*,j=*tetR*.

$$\frac{dm_i}{dt} = -m_i + \frac{\alpha}{(1 + p_j^n) + \alpha_0} \qquad (i)$$

$$\frac{dp_i}{dt} = -\beta(p_i - m_i) \qquad (ii)$$

With this model, is was possible to predict steady states and stable and unstable domains. The model was used to show that as the leakiness of fully-repressed promoters increases, the unstable steady state domain decreases, by plotting the stability boundary of α_0/α on a graph of the number of proteins in a cell against the ratio of protein and mRNA decay rates (Figure 2.6.5a, *left*). It was also shown that by increasing n, the Hill coefficient, the unstable domain becomes much larger. The authors also acknowledged the importance of accounting for the stochastic nature of molecular interactions and adapted their equations in accordance with Gillespie's method [Gillespie, 1977]. It was found that the model still demonstrated oscillations, however with large variability, which was a distinctive feature of the actual oscillatory network.

Gardener *et al.* also used coupled differential equations to model their genetic toggle switch (Figure 2.6.5b, *right*) which accounted for cooperative repression of transcription and repressor protein decay. The repressor protein concentrations are represented by u and v and their respective rates of synthesis by α_1 and α_2. The cooperativity of repression of the promoters upstream of u and v are represented by β and δ, respectively. By plotting a graph of the concentration of one repressor protein against the other, when the rate of change of u and v is zero, they were able to illustrate a graph having one unstable and two stable steady states (Figure 2.6.5b, *left*).



Reproduced with permission from [Gardner *et al.*, 2000] © Nature Publishing Group (www.nature.com).

Figure 2.6.5b: A toggle network with two stable steady states (*left*). A pair of differential equations representing the rate of change of two competing repressor proteins (*right*).

$$\frac{du}{dt} = \frac{\alpha_1}{1+v^{\beta}} - u \qquad (i)$$

$$\frac{dv}{dt} = \frac{\alpha_2}{1+u^{\alpha}} - v \qquad (ii)$$

As the nullclines (formed when the rate of change of the proteins is zero) intersected three times to form three steady state regions, owing to their sigmoidal shape which occurs when $\beta, \gamma > 1$, the authors inferred that cooperative repression of transcription is a requirement for bistability. Also, when the rates of change are not balanced, the nullclines do not intersect, therefore only a single stable steady state is formed. The model also indicated that the toggle switch had two stable basins of attraction (low state and high state). A state either side of the separatrix in Figure 2.6.5b will tend towards one of these two basins.

Becskei and Serrano also used differential equations, based on thermodynamic-kinetic principles, to model the rate of change of protein concentration in an unregulated network and self-regulated network, such as their synthetic autoregulatory network [Becskei and Serrano, 2000]. By applying a Taylor Expansion they linearised each of these differential equations around their steady states and thus produced equations reflecting the stability of each type of network. The stability equations produced showed that the autoregulatory network is more stable than the unregulated network for all realistic parameter values. They also showed that when the binding constant of the self-regulating repressor is very low, the stability of the autoregulatory network decreases, especially when the repressor protein has a short half-life. As expected, both of these factors contribute directly towards the self-regulation mechanism.

2.6.6 Natural Design Principles

Genome-scale databases are being composed which catalogue transcriptional components and their interactions. *E. Coli* has been well-studied, possibly more so than any other cell, in terms of its genes and transcription. The knowledge attained is being incorporated into relational databases [Salgado *et al.*, 2004], [Keseler *et al.*, 2005]. It has recently been predicted that between 20% and 25% of the network interactions in *E. coli* have been identified [Salgado *et al.*, 2004].

The availability of the data provided by these databases has allowed computational analysis to be performed on documented transcriptional networks in order to characterise prevalent network connection arrangements, termed *motifs*, with which nature has endowed *E. coli* [Shen-Orr *et al.*, 2002]. Three network motifs were found to occur in *E. coli* with a frequency of statistical significance when compared to their occurrences in randomised networks based on the same characteristics of *E. coli* networks (Figure 2.6.6).

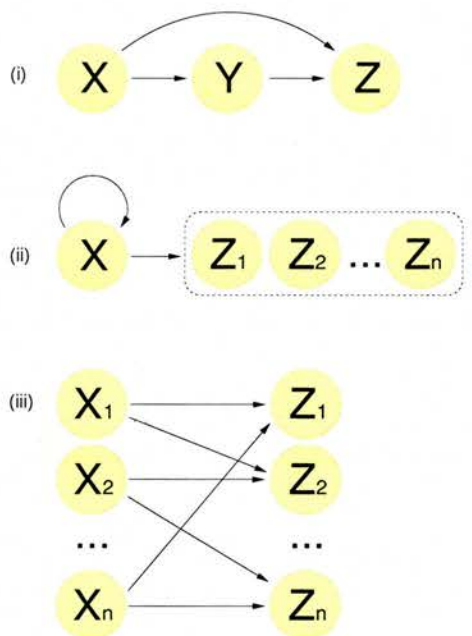


Figure 2.6.6: An illustration of three prominent network motifs found by Shen-Orr *et al.* through the analysis of *E. coli* transcriptional data. (i) A feedforward loop (FFL). Two transcription factors (TF), X and Y both regulate an operon, Z while X also regulates Y. (ii) A single input module (SIM). A single, often autoregulatory TF, X, controls a set of operons, $\{Z_1, ..., Z_n\}$. (iii) Dense overlapping regulons (DOR). Combinations of, usually different, inputs from a set of TFs, $\{X_1, ..., X_n\}$, regulate a set of operons, $\{Z_1, ..., Z_n\}$.

The first motif, a feedforward loop (FFL), was composed of three nodes where an operon is regulated by two *transcription factors* (TF) – proteins which bind to DNA and regulate gene expression – and where one TF also regulates the other TF. The majority (85%) of FFL networks were *coherant*, in that the inter-transcriptionary reg-

ulation was positive. Based on mathematical analysis, the authors propose that this motif may serve as a filter which rejects transient input signals, when both transcription factors are required to activate the output operon. A brief input signal will not allow a sufficient concentration of the inter-regulated TF (Y) to build up, whereas a sustained input signal will allow concentrations of both TFs (X and Y) to build up. This FFL mechanism also allows for rapid deactivation of the output operon as when the unregulated TF signal (X) is deactivated, the operon will become deactivated despite the lingering presence of the other TF (Y).

The second motif consisted of a single, often autoregulatory, TF which regulates a set of operons. These motifs, termed *single input modules*, were found in systems requiring a fixed proportion of proteins from the operons, such as in the construction of flagella. The third motif, termed *dense overlapping regions*, consists of a set of operons, each of which is controlled by, typically different, combinations of a set of input TFs. It was observed that the operons in each of the six instances of this motif found, share common biological functions.

However, Ma *et al.* questioned to what extent the results of network motif analysis is influenced by studying qualitatively incomplete databases [Ma *et al.*, 2004]. To investigate their question they merged data from RegulonDB [Salgado *et al.*, 2004], EcoCyc [Keseler *et al.*, 2005] and experimental literature to produce an integrated *E. coli* network. In the process of doing so, they also found many database inconsistencies of missing regulatory links. As found by Shen-Orr *et al.* they discovered FFLs to be highly represented, however a higher proportion of incoherent FFLs were characterised. They also found a tendency of FFLs to interact and produce a large motif cluster and that most genes are regulated by more than one FFL, or more complicated regulatory networks.

It is also interesting to consider network motifs which have not yet been found in *E. coli*. For example, Wall *et al.* were unable to find an instance of *inverse coupling* in 50 TFs which they studied [Wall *et al.*, 2004], where the effector gene expression is inversely proportional to regulator gene expression. According to earlier predictions, systems with high system *gain* – the ratio of the strength of an output signal to the strength of the input signal – are expected to have inverse coupling when under repressor control [Wall *et al.*, 2003]. The authors postulate that either the gain need not be

high for repressor control to function, or that the amount of inverse coupling required may be smaller than has been experimentally detectable.

2.6.7 Evolution *In Silice*

Principles of evolutionary design have been applied to computer simulation software [Foster, 2001] in order to optimise the rate constants of a cell-signalling pathway of fixed topology [Bray and Lay, 1994] and, more recently, to investigate the possible designs of genetic networks with specified functions [François and Hakim, 2004]. An evolutionary algorithm was developed which constructs a collection of networks of genes and reactions with randomised properties and subjects the collection to successive rounds of *growth* by mutation, where modified networks are added to the collection, and *selection*, where networks which obtain lower scores with a supplied fitness function are discarded. A representation of a network consisted of a set of genes, proteins (mRNAs were not explicitly represented for simplicity) and deterministic chemical reactions.

During the growth phase, mutations to be applied to existing networks were drawn from five categories: the degradation rate of a protein or the kinetic constant of a reaction may be chosen at random from existing constants and a random factor applied; a new protein-gene promoter interaction is created and a reaction for attachment; detachment and regulatory effect added with randomised constants; and post-transcriptional reactions of dimerisation or catalytic degradation. The authors note that further constraints on random mutations could be incorporated into their model to reflect physiological data for given cases. During a selection phase, each network in the collection is evaluated by integrating the set of coupled differential equations which represent the network reactions. A specific scoring function is supplied which ranks the networks and discards the lower-scoring half.

This method of directed search was applied to a bistable switch and oscillatory network. The scoring function for bistable switches was based on the concentrations of two competing proteins which were compared against desired values at given time points while a network was subjected to a pulse of a signal attempting to switch its state. At first, networks tended to contain *accessory* reactions which did not contribute towards the network's score, but rather were left over from previous evolutionary trials. Scoring functions were therefore adjusted to penalise larger reaction sets and a process

was created to systematically remove unnecessary reactions. In addition to finding the mutual inhibition motif, such as the one which characterises the genetic toggle switch of Gardener *et al.*, a number of variations were also found with greater frequency. The authors note that this may be a consequence of the fact that the number of reactions required in their representation for a mutual inhibition motif exceeds the number of reactions for alternative designs: therefore, introducing a selective bias. Interestingly, examples of some switches were found where their operation was dependent on post-transcriptional modifications. As a post-network design step, some chosen networks were simulated with stochastic dynamics applied to their rate equations to assess their resistance to noise. It was found that within this model, the networks which had been picked still functioned clearly as switches, even when only a few tens of regulatory particles were used to represent high concentrations.

Summary

This chapter has surveyed current work in the emerging field of Synthetic Biology, methods of DNA arrangement and approaches that are being used to aid the design of DNA programs. The next chapter will bring together elements from each of these areas by presenting software which automates the design of DNA programs, represented in the notation described in the previous chapter, and which supports rearrangement.

Chapter 3

Simulation Environment

This chapter describes and discusses the design of software which can explore a search space of candidate DNA program encodings and predict their behaviours through simulation.

3.1 Overview

The key purpose of the simulation software is to generate DNA programs (Section 1.7) which are predicted to meet a set of experimental criteria. It is in essence, a search tool, with access to a database of DNA site properties, which generates potential DNA program sequences in response to behavioural criteria supplied by an experiment (Figure 3.1). The genetic networks arising from these sequences are simulated to obtain tables of protein assays and DNA sequence information over time. These tables are used by the search tool to evaluate how well each sequence performs against the supplied, experimental criteria.

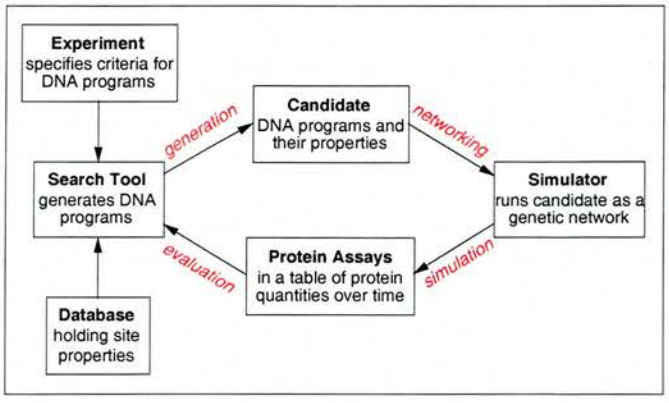


Figure 3.1: An overview of the simulation software's architecture. A search tool generates candidate programs aiming to fulfil supplied experimental criteria. These programs are simulated as genetic networks. The resulting simulation data is used by the search tool to assess the utility of the candidates against the supplied criteria. Thus, a cycle of generation and evaluation is formed.

3.2 Considerations

This section describes the considerations that were made when designing the search tool and simulation software. Section 3.2.1 discusses the practical limitations of making a *de novo* DNA program in a laboratory and the nature of the search space of DNA program compositions. The implications of supporting the simulation of rearrangeable DNA programs on the software machine model are discussed in Section 3.2.2. Section 3.2.3 considers the consequences of simulating DNA programs at different levels of abstraction in terms of computational efficiency and clarity of the simulation results. Finally, Section 3.2.4 highlights the importance of accounting for noise in gene expression and discusses the performance criteria of DNA programs.

3.2.1 Search Space

At present there are practical constraints on which genetic sites can be readily used to construct a *de novo* device in wetware. There will often be a selection of available sites, such as different promoters, stored in freezers. However, if one requires a genetic site with a new specific property (e.g. a promoter with an *E. coli* RNA polymerase rate of complex formation ten times lower than that of a bacteriophage T7 promoter) one can either choose an existing natural or synthetic component which most closely matches the criteria (e.g. a pML21 promoter), or alternatively use time-consuming techniques, such as site-directed mutagenesis, to generate a more exact match. The search strategy adopted here, in contrast to the generation of randomised, evolved networks [François and Hakim, 2004], is to perform searches over a database of *predefined* genetic components which represent readily available sites. Consequently, the search effort is focused on the generation of networks using these components (and their associated reactions), rather than compounding the network search space with the properties of the components themselves.

When searching for suitable candidates, if genetic site components in the database were to be randomly chosen and arranged, then the number of possibilities to be assessed would be infeasible, even for small numbers of sites. If ones considers a tape

made up of n sites and a database holding $|D|$ sites (${}^{|D|}C_n$ choices), allowing each site one of two orientations (2^n) and accounting for duplicate tapes that read the same when inverted ($\frac{1}{2}$), this produces the following number of possibilities:

$$2^{n-1}\frac{|D|!}{n!(|D|-n)!} \tag{3.2.1}$$

For example, if we were to choose 15 sites from a database of size 40, there would be over 6×10^{14} possibilities. Furthermore, Equation 3.2.1 does not account for the use of enzyme recognition sequences used in DNA manipulation. These sequences can be placed between sites and sometimes also include a range of options for some bases (Section 1.5.6) that can each be instantiated to one of a set of base pairs. For example, the restriction endonuclease *Bpu*10I recognises the pattern, 5'-CGTNAGC-3', where N can match any of the four nucleic acid bases. Therefore, if one considers including restriction endonuclease patterns, which can be placed in between any of the other types of site accounted for in Equation 3.2.1, this would further increase the equation's complexity.

Clearly, an exhaustive, brute-force approach towards traversing a search space in this way is impractical. Consequently, the search tool, which is to be explained in detail in Section 3.4, has been armed with methods to prune and order the search space with the aim of exploring areas most likely to be fruitful.

3.2.2 Simulation Machine Model

The aim of the software that simulates DNA programs is to predict a set of behavioural outcomes of a collection of one or more tapes created by the search tool as it traverses a search space directed by a set of criteria. The simulation model requires gene regulatory networks to be supported alongside DNA-arranging operations over nucleotide base sequences. Support is also required for the dynamic rearrangement of the regulatory networks during the course of the simulation, directed by protein interactions with arbitrary base sequences.

Such a model not only requires state, but the interpretation of that state drives the time-changing properties of the model, including the structure of its connections. This can be described as reprogramming. Consequently, the use of more structurally sta-

tic descriptions, such as the ordinary differential equations used to describe a genetic toggle switch [Gardener *et al.*, 2000], do not lend themselves to these requirements [Webb and White, 2005]. Furthermore, owing to DNA arrangement operations, a model may be reprogrammed multiple times during the course of a simulation and it may not be possible to directly understand the behaviour exhibited by a model in terms of its initial composition. Therefore, it is especially important that the rearrangements of a model and the events leading to those rearrangements are transparent: both to a human observer and to the automated search tool which requires meta-information about the DNA arrangements in order to evaluate some criteria. To address this, the basis of each model is described using a symbolic notation of DNA sites, as described in Section 1.7. However, in order to enable the stochastic simulation of cellular reactions [Turner *et al.*, 2004], strings in this symbolic notation were mapped to networks of interacting components (Section 3.7.2). This enabled simulated molecules to be passed stochastically, as messages, over graphs between genetic sites and the host cell. This approach, in general, is more computationally efficient than adopting a more detailed, spatial representation to encode the location and movement of each molecule [Le Novère and Shimizu, 2001], [Lipkow *et al.*, 2005]. This message-based model also naturally accommodates the representation of individual molecules, in contrast to the approach of forming a graph of reactions, between species of molecules and their concentrations [Meir *et al.*, 2002].

3.2.3 Clarity and Efficiency

A large number of candidates can be expected to be simulated when traversing a search space even when considering a small number of sites, owing to the combinatorial nature of DNA arrangements. Consequently, the time taken for each simulation is an important practical issue. Simulating typically several thousand base pair plasmids at the lowest level of abstraction required, that of the nucleotide base sequences and the protein interactions (requiring, for example, extensive pattern matching) is significantly more expensive than adopting a hybrid approach where only recognition sequences are modelled as nucleotide bases and other sites, such as promoters, are modelled at a higher, more behavioural level of abstraction.

A limitation of this approach is that it does not allow for those potentially complex scenarios where DNA-arranging enzymes are able to modify DNA *inside* a functional site such as a protein encoding. For example, if a protein encoding were to be digested part way along its length, information would need to be provided as to how effectively the resulting protein would fold and function. However, by following this approach, the simulation results can be more readily understandable (and consequently verifiable) as they are expressed in terms of the sites in the notation presented, along with some potential linking sequences in between. For example, in Section 3.3, the search tool was instructed to compose a DNA program which produces a stable level of Cro protein. One resulting tape was expressed concisely as, $O_{R2}P_RX_{cro}T_{R1}$. Each of the four components that make up this tape represent well-characterised sites in bacteriophage λ, and a reader familiar with their names can identify the associated functionalities. P_R is an effective promoter, so Cro will be synthesised. When Cro is attached to O_{R2} this will cause the downregulation of P_R, and therefore Cro regulates its own production.

3.2.4 Variation of Gene Expression

Levels of protein production in *E. coli* cells can vary widely [McAdams and Arkin, 1999]. Influences include many parameters which are not fully within the control of the experimenter. For example, protein production of a candidate tape can be affected by the cells' changing environmental conditions and considerable variation can be seen both between cells, and within one cell over time [Elowitz *et al.*, 2002], depending on their stage of growth and many other factors. Living cells are certainly not simple deterministic automata. For these reasons, it is important to account for the stochastic nature of those reactions in the simulation model that can cause state changes [Rao *et al.*, 2002]. In doing so, the search tool can judge how *reliably* (over a number of simulation runs with different pseudo-random seeds) a candidate meets a set of criteria.

Three key performance criteria of simple genetic networks: *stability, robustness* and *responsiveness* have been proposed [Wall *et al.*, 2004]. The *stability* of a network to resist transient disturbances is accounted for here through the modelling of stochastic interactions, typically using less than 100 particles per signal and by performing multiple simulation runs to assess consistency of behaviour. All reaction parameters associated with sites are used in probabilistic equations. Thus, a parameter which is not *robust*, in that changes in the parameter cause changes in the network's ability,

will contribute towards a loss of consistency when a network is evaluated over a set of runs. An alternative approach is to sweep the value of each parameter incrementally and assess the affect on a network's ability [Meir *et al.*, 2002]. In terms of this software, both the parameter values and network compositions are important aspects influencing the outcome of a search. Furthermore, modelling reactions of particles stochastically is computationally expensive relative to other methods of evaluation, such as by integrating ordinary differential equations. Given this, and that the objective here, is to perform a complete traversal of a search space for networks of components based on predefined properties, by performing incremental sweeps of parameters, each sweep would require its own network search thus exploding the size of the search space. Otherwise, if sweeps were performed post-search, the search would have to have been based on a single set of predefined parameters, which would influence, and therefore limit, the choices of networks selected during search. The third performance criterion, *responsiveness*, which describes a system's ability to settle into a new steady state following an environmental change, can be accounted for by evaluating the levels of protein concentrations at time points after a simulated cell has been subjected to an external signal.

3.3 Software Summary

A search tool, called the "DNA Program Generator", was created to explore the space of potential synthetic DNA programs obeying specified criteria. Criteria are used to impose constraints on the compositions of DNA program candidates and to specify the desired behaviours of candidates. Constraints influence how DNA sites that are used to compose candidates are chosen from the database. The behaviour of a candidate is assessed in one or more experiments, called *trials*, which direct the simulation of each candidate in different environmental conditions and test how closely a candidate behaves in a desired manner: In each trial, molecules may be added to the cell, measured and compared over time points or intervals to assess conformity with a set of rules.

```
Trial:  flat

Send 0.01 units of rnap from 0:00 to 12:00.
   Check cro at 04:00 is more than 0.
   Check cro at 12:00 is more than 0.
   Check cro from 04:00 to 12:00 is stable.
```

Figure 3.3a: A short example of candidate criteria for the DNA Program Generator. A trial, labelled "flat", states that a small amount of RNA polymerase is to be produced for 12 hours, that some Cro should be detectable after 4 and 12 hours and that the amount of Cro during the interval should also be stable.

For example, the criteria in Figure 3.3a show a transcription protein, Rnap, being produced and a set of three rules requiring the measurement of a protein, Cro.

```
> operator oR2
>   up(cI, 0.3, 0.02, pRM, 25.0)
>   down(cro, 0.3, 0.02, pR, 0.90)

ctaacaccgtgcgtgttga
```

Figure 3.3b: A sample operator, O_{R2}, in the site database. If a CI or Cro protein is nearby during a simulated time unit, the probability of attachment is 0.3 and the probability of decay is 0.02. While Cro is attached, a nearby pR site would be down regulated by 90%. Similarly, CI causes up regulation of pRM.

The search tool is also provided with a database of genetic sites, their properties and relations (Figure 3.3b). The database holds information on operators, promoters, protein encodings, terminators, plasmids and DNA-arranging enzymes, which is used to construct candidates that may be able to fulfil a given set of criteria. One of the candidates generated is shown in Figure 3.3c.

```
i:   oR2(+)pR(+)cro(+)tR1(+)pBR322(+)

ii:  O_R2 P_R X_cro T_R1
```

Figure 3.3c: A sample candidate expressed in plain text and notation. (i) A textual representation with parenthesised plus and minus signs representing sense and anti-sense directions, respectively. (ii) A representation in notation, without reference to a plasmid backbone.

The "DNA Program Simulator", accepts such a candidate from the search tool, constructs reaction networks of the candidate's sites and simulates its behaviour over time. Although some reactions are simulated stochastically, static graphical illustrations can be produced automatically that capture changes in state of a candidate as molecules attach and detach over simulated time (Figure 3.3d).

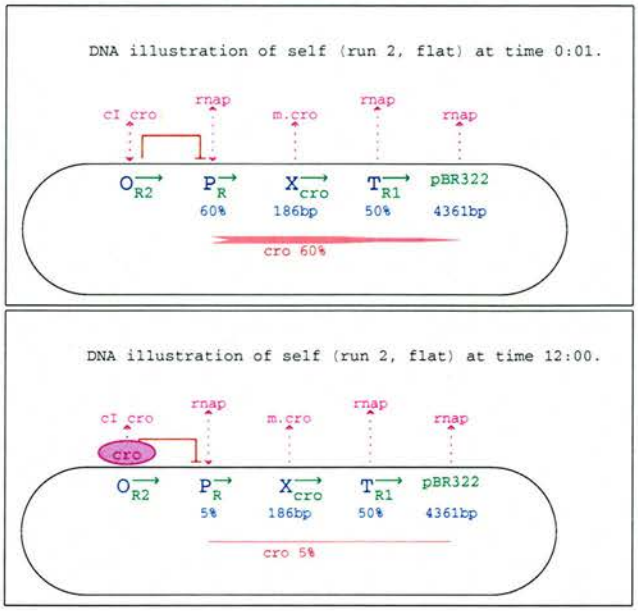
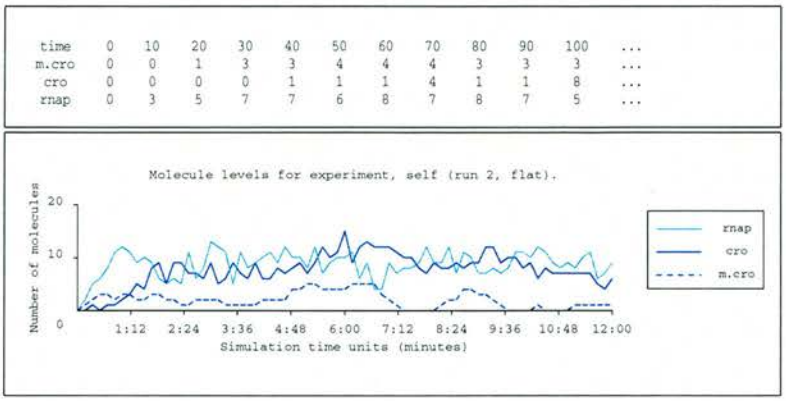


Figure 3.3d: Graphical representations of a sample candidate's simulation at the beginning (above) and end (below) of a run, generated by the DNA Program Simulator. Sites are shown as blue capital letters with green subscripts for their types and an arrow indicating their orientations. Per unit time, a nearby RNA polymerase may attach to the P_R site with a chance of 60% (above). A red arrow extends from underneath the promoter in the direction of its orientation, the thickness of the arrow being proportional to the anticipated *average* strength of transcription. The thickness of the transcription arrow reduces as it passes through the terminator, T_{R1}. Underneath the transcription arrow, the *cro* gene is shown to be expressed along with the strength of transcription. The dashed magenta arrows show potential molecular attachments and detachments. Operator O_{R2} has a brown arrow directed at P_R indicating a regulatory relationship. If a Cro molecule attaches itself (below), then the effectiveness of the promoter can be seen to decrease.

Candidates are simulated, under the conditions of each trial, for a specified number of instances, called *runs*. The ordinal of each run is used to seed a linear congruential pseudo-random number generator [Lehmer, 1949] before each simulation begins. This provides an indication of the consistency of the behaviour of the candidate with respect to those aspects simulated stochastically (Section 3.2.4). The stream of pseudo-random numbers produced by generator is identical for each seed, therefore, the outcome of a particular run of a trial can be repeated precisely. For each trial's run, a table of protein assays – measurements of the amounts of different proteins in a simulated cell – is recorded over time. This data is then used by the search tool to evaluate the fitness of the candidate against the criteria. For example, the table in Figure 3.3e shows the protein assays of the example candidate during the first run of its "flat" trial.



time	0	10	20	30	40	50	60	70	80	90	100	...
m.cro	0	0	1	3	3	4	4	4	3	3	3	...
cro	0	0	0	0	1	1	1	4	1	1	8	...
rnap	0	3	5	7	7	6	8	7	8	7	5	...

Figure 3.3e: Protein assays during a simulation of the example candidate. The first entries of a spreadsheet of protein quantities for the first run are shown above for Rnap, Cro and mRNA Cro (m.cro). The graphical plot output of the DNA Program Simulator is shown below. The amount of Rnap (cyan) is consistent throughout. The amount of Cro (dark blue) and its mRNA (dashed) show more fluctuation but are also stable.

Each spreadsheet is evaluated against the given criteria and the candidate is awarded marks for each rule that it satisfies, weighted by any importance specifiers. Candidates can then be ranked by fitness, i.e. their accumulated scores. Those candidates having the same score can be further sorted by either the lengths of their DNA program sequences (in base pairs) or the number of DNA sites used from the database. Each of these two metrics provides an indication of the cost of DNA synthesis in a laboratory. A summary of the rules satisfied by each candidate is also produced to justify the scores awarded, as shown in Figure 3.3f.

```
Trial:  flat

   3x cro at 04:00 is more than 0 in runs 1, 2 and 3.
   3x cro at 12:00 is more than 0 in runs 1, 2 and 3.
   2x cro from 04:00 to 12:00 is stable in runs 1 and 3.
```

Figure 3.3f: A summary of rules satisfied by the example candidate over three runs. The protein Cro was detected in all runs at 04:00 and 12:00 and was stable in between these times in two out of the three runs.

This candidate was ranked first based on the initial criteria and is an example of self-regulation. The Cro protein is able to bind to the operator O_{R2} which will reduce the level of its own production. Thus, if the amount of Cro is high it is likely that further mRNA Cro production will be inhibited, whereas if the amount of Cro is low, its production is likely to be uninhibited. In contrast, an unregulated candidate produced by the generator is shown in Figure 3.3g.

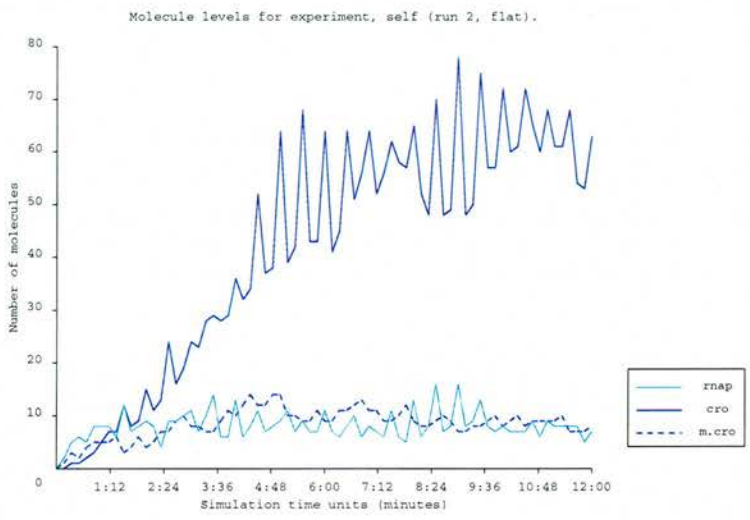


Figure 3.3g: The graphical plot output for one of the runs of the unregulated candidate, $P_R X_{cro} T_{R1}$. The amount of Cro mRNA (dashed) is considerably higher than in the case of the regulated example (Figure 3.3e). In the absence of an operator, the rate at which the amount of Cro (dark blue) increases is limited only by the decay rate of Cro (and its mRNA). Consequently, the amount of Cro climbs much higher.

3.4 Generation of Candidates

The search tool is responsible for composing *de novo* plasmids by assembling components from a database of sequences. It is also able to assess the results of how such plasmids behave, in terms of their simulated output signals when subjected to a set of input criteria. This section describes how the generator constructs potential candidates for evaluation given a database and a set of criteria.

The graphical user interface of the generator is presented in Appendix A.1 and the syntax of the criteria in Appendix A.5. Format details of databases describing DNA sites and their properties are provided in Appendix A.6. The software, examples, results and source code are available on-line at, "http://blenkiron.com/phd".

3.4.1 Encoding Sets

Firstly, the generator gathers a set of "core" protein encodings, X^{core}, from the database (each of type, protein) which are considered vital to the success of any candidate.

Encodings from this set will be used in every candidate considered. If the level of expression of a protein, q, is used in a rule ($q \in$ expression) and the protein is not provided externally ($q \notin$ command)[1], then this protein's encoding is considered essential to be on the tape, otherwise the rule would be redundant. The criteria may also force the use of a number of proteins (the include set)[2].

$$q \in X^{core} \iff (q \text{ is} \cdot \text{a protein}) \wedge ((q \in \text{expression} \wedge q \notin \text{command}) \vee (q \in \text{include}))$$

Next, the generator creates an "optional" set of protein encodings, X^{opt}, which are not in the core set, which *may* have an effect on the behaviour of the candidates. The criteria may also explicitly specify the optional proteins to be considered (the specify set) and may exclude proteins from consideration (the exclude set).

$$q \in X^{opt} \iff (q, r \text{ is} \cdot \text{a protein}) \wedge (q \notin X^{core} \cup \text{exclude}) \wedge (\overline{\exists} r \in \text{specify} \vee q \in \text{specify})$$

In addition, enzymes that can linearise plasmids must be used in conjunction with enzymes that can circularise, otherwise fragments of DNA that could not be copied by a bacterium would be produced (Section 1.5.5).

Sets of protein encodings for the composition of a candidate can be generated from the core and optional sets by considering each of the possible ways of choosing sites from X^{opt}, and then combining the result with X^{core}. The number of optional sites chosen is referred to as a protein *stage*. At a given protein stage, the number of sets for a candidate cannot exceed ${}^{|X^{opt}|}C_{stage}$. The criteria may also place a limit on the maximum number of encodings allowed in each tape[3].

3.4.2 Operator Sets

Operator sites are also drawn from the database (each of type, operator) and have core and optional sets similar to those of protein encodings. The core set is simply

[1]Protein and mRNA signals can be sent to a cell at time points and over time periods to increase protein amounts. The `Send` command can be used in evaluation criteria to send signals (Section A.1.4.1).

[2]The generated constraints on the sets of proteins considered for the "core" protein set can be overridden through the use of three commands. The `include` command forces the use of a set of encodings (Section A.1.2.1), the `specify` command limits consideration of choices to a set of encodings (Section A.1.2.2), and the `exclude` command removes a set of encodings from consideration (Section A.1.2.3).

[3]The maximum number of protein encodings allowed in generated candidates can be specified using a `maximum protein` command in a set of criteria (Section A.1.3.3).

composed of those sets that the criteria explicitly include (the include set)[4].

$$q \in O^{core} \iff (q \text{ is} \cdot \text{a operator}) \wedge (q \in \text{include})$$

When creating the optional operator set one avoids including operator sites that could not have an influence on the outcome of the candidate's simulated behaviour. If one considers the proteins provided externally with a chosen set of encodings for the composition of a candidate, X, then all the proteins that could be expressed during a simulation can be determined. Given this, those operators that have an affinity for at least one of these proteins can be added to the optional set. In a manner similar to protein encodings, the criteria may force the exclusion (the exclude set) and overall specification (the specify set) of which operators to use.

$$q \in O^{opt} \iff (q \text{ is} \cdot \text{a operator}) \wedge (q \notin O^{core} \cup \text{exclude}) \wedge$$

$$((\exists s \in \text{command}) \vee (\exists s \in X) \wedge s \text{ has} \cdot \text{affinity} \cdot \text{for } q) \wedge (\overline{\exists} r \in \text{specify} \vee q \in \text{specify})$$

As in the case of protein encodings, operators have stages where each stage corresponds to the number of optional operators chosen from O^{opt}.

3.4.3 Encoding and Terminator Layout

No restrictions are placed on the possible permutations or orientations of a given encoding set, X, as it is laid out on a candidate tape. Duplicates that read the same in opposite directions are also allowed. Combined permutation and orientation choices can be represented by an integer function, γ, of the form:

$$\gamma = \{i \mapsto (j,k)\} \ : \ 0 \leq i,j < |X|,\ k \in \{L,R\}$$

A set of mappings, $\Gamma = \{\gamma_0, \ldots, \gamma_{n-1}\}$, holds $|X|!$ permutations and $2^{|X|}$ orientations:

$$|\Gamma| = 2^{|X|}(|X|!)$$

[4]Constraints on operator sets can also be affected by the use of `include`, `specify` and `exclude` commands in criteria.

Each protein encoding, $x_i \in X$, has a terminator, t_0, appended to it upstream[5] and such a pair is represented by $g_i \in G$. There is also a special protein encoding, x_{blank}, which has the purpose of adding a space between the locations either side of it without actually encoding for a protein. When $x_i = x_{blank}$, a terminator is omitted to allow a space without hindering transcription. The resulting tape layouts can be represented in general as:

$$g_{\gamma_j(0)} \quad g_{\gamma_j(1)} \quad g_{\gamma_j(2)} \quad \cdots \quad g_{\gamma_j(n-1)} \quad : 0 \leq j < |\Gamma|,\ n = |X|$$

3.4.4 Operator and Promoter Layout

Given a chosen set of operators, O, each operator, $o_i \in O$, has a promoter attached downstream, and is represented by $s_i \in S$. As we allowed for all orders and orientations of protein encodings, including inverses (Section 3.2.1), we consider inserting these operators into sites to the left of each of the protein encodings. Potential insertion sites are shown by $i_{\{0,\ldots,n-1\}}$:

$$i_0 \quad g_{\gamma_j(0)} \quad i_1 \quad g_{\gamma_j(1)} \quad i_2 \quad g_{\gamma_j(2)} \quad \cdots \quad i_{n-1} \quad g_{\gamma_j(n-1)} \quad : 0 \leq j < |\Gamma|,\ n = |X|$$

When limiting the choice of the locations and orientations of the operators from a given operator set, there are a number of aspects that should be considered to provide coverage of possible regulatory networks. Firstly, it should be possible for each operator (with a promoter) to be immediately upstream of each encoding so that it can regulate it. Secondly, it should be possible for each pair of operators to be next to each other so that combined operator effects can occur. In addition, pairs of operators are also required to point in opposite directions. Thirdly, each operator is given an opportunity to be positioned in the special location, i_0, which is next to a *single* protein encoding only, rather than in between two protein encodings. An operator layout mapping function (Section A.4), ω, is able to satisfy these requirements and is composed of $2|X|$ ordered sets:

$$\omega = x \mapsto ([y|null], [L|R]) : 0 \leq x < 2|X|,\ 0 \leq y < |O|, |O| \leq |X|.$$

[5]The type of default terminator can be specified in the criteria using a `default terminator` command (Section A.1.3.1).

Operators will be provided with a downstream promoter that they are able to regulate, otherwise a default promoter will be used[6]. Combined operators and promoters are represented by s_i. The resulting layouts can be represented, in general, given $0 \leq j < |\Gamma|$, $n = |X|$, $0 \leq k < 2|X|$, as:

$$s_{\omega_k(0)} \quad g_{\gamma_j(0)} \quad s_{\omega_k(1)} \quad g_{\gamma_j(1)} \quad s_{\omega_k(2)} \quad g_{\gamma_j(2)} \quad \cdots \quad s_{\omega_k(n-1)} \quad g_{\gamma_j(n-1)}$$

3.5 Pruning by Expression

When considering plasmids that do not spontaneously rearrange their DNA after entry into a host cell without prompting by any external or environmental signal, the search space can be reduced by pruning some of the generated encoding-operator layouts. In some instances, before the simulation begins, it is possible to calculate whether a layout cannot fulfil some of the rules in the criteria. These calculations require a small number of instructions to be processed per rule, whereas simulation, in comparison, is computationally expensive.

3.5.1 Rules

A pre-simulation assessment of the layouts produces a table of scores to indicate how well each layout conforms to a subset of the rules in the criteria as shown in Table 3.5.1. This subset consists of all types of rule except those that involve the prediction of potential arrangements of fragments of DNA, which requires simulation. Based on these scores, there are a number of ways of deciding which candidates should be simulated.

[6] A `default promoter` command in the criteria can set the promoter that will be attached to operators by default (Section A.1.3.1).

Rule Template	Liberal Assessment
x `is less than` y	$x_{min} < y_{max}$
x `is more than` y	$x_{max} > y_{min}$
x `is much less than` y	$x_{min}(1+G) < y_{max}$
x `is much more than` y	$x_{max} > y_{min}(1+G)$
x `is less than or equal to` y	$x_{min} \leq y_{max}$
x `is more than or equal to` y	$x_{max} \geq y_{min}$
x `is equal to` y	$x_{min} \leq y_{max} \wedge x_{max} \leq y_{min}$
x `is approximately equal to` y	$x_{min} \leq y_{max}(1+S) \wedge x_{max}(1+S) \leq y_{min}$ $\vee\ x_{min}+S \leq y_{max} \wedge x_{max} \leq y_{min}+S$
x `is increasing from` t_0 `to` t_1	$x^{t_0}_{min}(1+G) < x^{t_1}_{max}$
x `is decreasing from` t_0 `to` t_1	$x^{t_1}_{min}(1+G) < x^{t_0}_{max}$
x `is stable from` t_0 `to` t_1	$x^{t_0}_{min}(1+G) > x^{t_1}_{max} \wedge x^{t_1}_{min}(1+G) > x^{t_0}_{max}$

Table 3.5.1: The subset of rules that is used to predict the maximum possible score of a candidate before it is simulated. Some rules refer to specific time points, others span intervals. Variables x and y may be instantiated with the calculated upper and lower bounds (Section 3.5.3) of a protein assay or a constant number. The proportions S and G are the threshold bounds `slightly` and `greatly` which can be configured in the criteria of an experiment (Sections A.1.3.7 and A.1.3.7). A rule will be assessed to be true if there is a *possibility* that the candidate may fulfil it. This is because the purpose of pre-simulation scoring is to exclude candidates that cannot attain a criteria-specific minimum score, while not excluding those that have the potential to do so.

3.5.2 Modes

There are three "pruning" modes which determine how candidates are removed from being considered for simulation based on their pre-simulation scores[7]. In all cases, if a limit has been set for the maximum number of candidates to consider simultaneously[8], then lower-scoring candidates will be replaced by higher-scoring candidates if this limit is exceeded. The first mode does not attempt to reduce the search space. The second mode discards layouts with zero score if at least one other layout has a non-zero score. The third mode discards all layouts other than those with a potential maximum score.

[7]One of the three pruning modes: `none`, `worst` or `suboptimal` can be specified with criteria commands (Section A.1.3.6).

[8]The `maximum pool` command allows a limit to be imposed on the number of candidates to be considered at once (Section A.1.3.4).

When a gene set does not contain any DNA-arranging enzymes, one can consider the DNA state to be stable until after the time such an enzyme may be introduced. When a gene set does contain arranging enzymes, we can consider the DNA to be stable until such time that a protein is provided that may cause the direct or indirect production of an arranging enzyme.

3.5.3 Expression Bounds

Given a layout of proteins (their encodings), promoters, operators, terminators and their orientations, the upper and lower bounds on the expression of each gene can be determined. For each operator, we can consider all of the possible regulatory effects on nearby promoters when available proteins are attached and unattached. Consequently, we can calculate upper and lower bounds on the affinities of promoters for RNA polymerase. Based on this, one can assign a minimum and maximum transcription-causing effectiveness value to each promoter. Next, we consider those promoters upstream of each protein encoding which are similarly oriented to calculate bounds on the production levels of proteins which are directly affected by the upstream effectiveness values. If more than one promoter is oriented towards a protein encoding, the effectiveness values for that encoding accumulate as this corresponds to transcription beginning from two locations at once. When a promoter is oriented towards an encoding with a termination site in between, the pass-through transcription reduction caused by the termination site is accounted for by reducing the effectiveness values with the mean proportional blocking effect of the terminator before it is accumulated by the encoding.

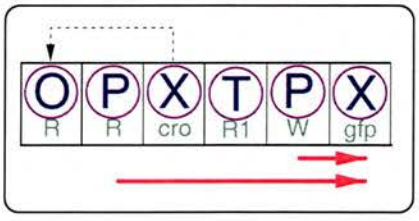


Figure 3.5.3: A graphical representation of the DNA program, $\mathbf{O_R P_R X_{cro} T_{R1} P_W X_{gfp}}$. $\Phi = (\mathbf{P_R}, 0.6), (\mathbf{P_W}, 0.1), (\mathbf{T_{R1}}, 0.5), (\mathbf{O_R}, \mathrm{M_{cro}}, \mathbf{P_R}, -0.95)$.

For example, consider the layout of the DNA program shown in Figure 3.5.3. The maximum effectiveness of the promoter, $\mathbf{P_R}$, is its base value of *0.6*. When a Cro protein attaches to $\mathbf{O_R}$, this will downregulate the promoter by a factor of *0.95* which results in a minimum effectiveness value of *0.03* (0.6 $\times$ (1-0.95)). The effectiveness of the second promoter, $\mathbf{P_W}$, cannot be altered and has both an minimum and maximum value of *0.1*.

Next, we can calculate the upper and lower bounds on the expression of each protein. X_{cro} is downstream of only one promoter, P_R, and therefore it acquires its expression bounds directly from that promoter's effectiveness limits, *0.03* to *0.6*. However, the transcription of X_{gfp} may begin from P_W and P_R. The terminator, T_{R1}, will block half of the RNA polymerase molecules which attach at P_R. Therefore, the effectiveness limits of P_R towards the transcription of X_{gfp} are *0.015* to *0.3* (dividing 0.03 and 0.6 by two). Finally, to calculate the expression bounds of X_{gfp}, one can sum the effectiveness limits of P_W and P_R to produce, *0.115* to *0.4*.

3.5.4 Prediction

Next, for each trial, the rules that occur while the DNA of a layout is stable are collected together (Section 3.5.1) and their ranges are tested against the list of protein limits and accumulate a weighted score for each layout. When it is uncertain whether a rule will be passed, it is given a score so as not to remove potentially good candidates.

```
1 Check gfp at 1:00 is more than bfp at 1:00.
2 Check bfp from 1:00 to 2:00 is decreasing.
3 Check bfp at 2:00 is approximately equal to 0.
```

Figure 3.5.4: Example rules used for score prediction based on assays of proteins, BFP and GFP.

For example, one can consider the set of rules in Figure 3.5.4 when the sample expression bounds of one gene, *bfp*, are *0.2* to *0.4* and another gene, *gfp*, has bounds of *0.0* to *0.7*. The first rule can be satisfied as the upper bound of *gfp* (0.7) exceeds the lower bound of *bfp* (0.2). It is possible that this rule may not be satisfied in an actual simulation run (e.g. if *gfp* is not expressed) – however, rules that *might* be satisfied will be passed during pruning. The second rule can also be satisfied as *bfp* expression could potentially drop by a factor of two (0.4 to 0.2). However, the third rule cannot be satisfied, as *bfp* will be continually expressed (0.2).

3.6 Recognition Sequences

Once a tape has been constructed with operators, promoters, encodings and terminators, recognition sequences can be added for DNA-arranging enzymes. Although it would be possible to consider using recognition sequences *inside* the four site types

listed above, this would complicate their simulation as these sites could no longer be considered atomic for the duration of a run. Instead, recognition sequences are encoded at a lower level of abstraction during simulation using nucleotide bases. This allows recognition sequences to support manipulation by DNA-arranging enzymes.

3.6.1 Sequence Sets

After determining a set of proteins that arrange DNA and may be present during a simulation, we can construct a set of sequences that would be recognised by these proteins. From this, we can also construct a set of compatible sequences, Z, for enzymes that break DNA. Like operators and encodings, there are also recognition sequence stages that correspond to the number of sequences used. At each stage, compatible sequences are chosen from each set, which contains $|Z|^{stage}$ possibilities. Only sets of sequences that are able to form potential interactions with each other are considered, as otherwise the only operation that could be performed would be isolated, such as a single digestion. This can significantly reduce the number of possibilities considered as most recognition sequences are able to interact with a small number of others, relative to the large number of DNA-arranging enzymes.

For example, consider a set of proteins containing three enzymes and their recognition sequences, *Xba*I (5'-T↓CTAG↑A-3'), *Nhe*I (5'-G↓CTAG↑C-3') and *Bpu*10I (5'-CC↓TNA↑GC-3'). *Xba*I and *Nhe*I can create the same cohesive end, 3'-CTAG-5'. *Bpu*10I can produce two pairs of compatible cohesive ends, 3'-TAA-5' and 3'-TTA-5', and TCA and TGA. Therefore, the set of compatible sequences contains three entries and each entry has two associated oligos (of type R). This set can be described as, Z = { 3'-CTAG-5' (R_{TCTAGA}, R_{CCTAGG}) }, { 3'-TAA-5', 3'-TTA-5' ($R_{CCTAAGC}$, $R_{CCTTAGC}$) }, { 3'-TCA-5', 3'-TGA-5' ($R_{CCTCAGC}$, $R_{CCTGAGC}$) }.

3.6.2 Sequence Locations

Recognition sequences can be placed in between any of the other sites of a tape. For each item in the current encoding set, X, there is also a promoter, operator and terminator ($4|X|$ sites). To reduce the search space size, the resolution of how they are positioned can be specified in the criteria. The step size, σ, determines the granularity, in terms of sites, and the distance, δ, specifies the number of sites to place between pairs of compatible recognition sequences. Given the recognition sequence stage de-

termines the number of sequences used; an upper bound on the number of placements per stage is $(\frac{4|X|-\delta}{\sigma}+1)^{stage}$. However, requiring all sequences to have potential interactions with each other can significantly reduce this.

$R_{TCTAGA}O_RP_RX_{bfp}T_{R1}R_{TCTAGA}O_{lac}P_{lac}X_{cI}T_{R1}$ $R_{TCTAGA}O_RP_RX_{bfp}T_{R1}O_{lac}P_{lac}X_{cI}T_{R1}R_{TCTAGA}$ $O_RP_RX_{bfp}T_{R1}R_{TCTAGA}O_{lac}P_{lac}X_{cI}T_{R1}R_{TCTAGA}$
$R_{CCTAGG}O_RP_RX_{bfp}T_{R1}R_{CCTAGG}O_{lac}P_{lac}X_{cI}T_{R1}$ $R_{CCTAGG}O_RP_RX_{bfp}T_{R1}O_{lac}P_{lac}X_{cI}T_{R1}R_{CCTAGG}$ $O_RP_RX_{bfp}T_{R1}R_{CCTAGG}O_{lac}P_{lac}X_{cI}T_{R1}R_{CCTAGG}$
$R_{TCTAGA}O_RP_RX_{bfp}T_{R1}R_{CCTAGG}O_{lac}P_{lac}X_{cI}T_{R1}$ $R_{TCTAGA}O_RP_RX_{bfp}T_{R1}O_{lac}P_{lac}X_{cI}T_{R1}R_{CCTAGG}$ $O_RP_RX_{bfp}T_{R1}R_{TCTAGA}O_{lac}P_{lac}X_{cI}T_{R1}R_{CCTAGG}$

Table 3.6.2: The generation of tapes with additional recognition sequences. A tape, a sequence layout and a compatible mid-sequence set were combined to form nine new tapes. In the top group, the sequence, R_{TCTAGA}, is used twice. Similarly, in the middle group, the sequence, R_{CCTAGG}, is also used twice. The bottom group uses one of each of these two sequences. The order of the sequences is not permuted as the order of the other types of site is permuted instead.

For example, consider the tape, $O_RP_RX_{bfp}T_{R1}O_{lac}P_{lac}X_{cI}T_{R1}$. When σ is set to a granularity of four sites and δ is set to a distance of four sites, there are three possible insertion locations for sequences: * $O_RP_RX_{bfp}T_{R1}$ * $O_{lac}P_{lac}X_{cI}T_{R1}$ *, each location being marked by an asterisk. Table 3.6.2 lists the results of combining this tape and these parameters with a small set of Z = { 3'-CTAG-5' (R_{TCTAGA}, R_{CCTAGG}) }, during a second sequence stage (which considers two sequence choices at a time).

3.6.3 Connecting to the Simulator

```
# BiSim Configuration Script (for Arrange DNA Model)
# Automatically Generated by DNA Search Tool

<Arrange>

experiment    Example_(run_1-_flat)
cell          synthesis
tape          oR2(+)pR(+)cro(+)tR1(+)pBR322(+)plasmid(+)
restrict      ,
commands      rnap@0..1440:1%
stickiness    ,
ligases       dlig
proximity     100%
circular      10
read_speed    50
labels        gfp=green,bfp=blue,bpu10I=red

              ...

<Network>     steps 1440
<Random>      seed 1

<pBR322>      $ Plasmid(pBR322,4361bp)
<oR2>         $ Operator(oR2,cI;30%<attract>;
                2%<decay>;pRM;2500%<effect>|...
                cro;30%<attract>;2%<decay>;pR;9%<effect>)
<cro>         $ Encoding(cro,186bp,5%<decay>)
<pT7>         $ Promoter(pT7,90%<attract>)
<tR1>         $ Terminator(tR1,50%<effect>)
<synthesis>   $ Synthesis(cell,5%<mRNA_decay>;...
                20%<mRNA_trans>;cro=3%;rnap=30%)
```

Figure 3.6.3: An example simulation file produced by the DNA Program Generator. Each of the fields is described in turn: *experiment*, title of the experiment, run number and trial name; *cell*, module name of cell; *tape*, candidate tape; *restrict*, sequences, affinities and effects of DNA-arranging enzymes; *commands*, proteins injected into simulation; *stickiness*, affinities of cohesive ends; *ligases*, proteins rejoining DNA; *proximity*, per unit time, chance of a ligase being next to DNA; *circular*, how much more likely DNA is to circularise than to join separate stands; *read_speed*, number of codons an RNA polymerase reads per unit time; *labels*, specifies colours for proteins; *steps*, number of simulation steps; *seed*, random seed, varies by the run; *pBR322*, plasmid node, with size; *cro*, encoding node, with size and protein decay rate; *pT7*, promoter node, with RNA polymerase affinity; *tR1*, terminator node, with effectiveness; *synthesis*, cell module with protein decay rates.

Once a candidate has been generated, the simulator is invoked with its tape layout, external commands, properties and relations of the sites, proteins and sequences. A simple example covering some of the possible settings and nodes is shown in Figure 3.6.3. The simulator, as described next, will reply with a data file of protein amounts over time to be used for evaluating the criteria's rules.

3.7 Simulation Approach

3.7.1 The BioSim Platform

The BioSim platform is a discrete-time, event-driven simulation platform with which the DNA Program Simulator was created. The platform's purpose is to explore models of peer-to-peer networks composed of interacting devices (e.g. operators, promoters, encodings). Devices' state and functionality can be composed in a modular fashion

by plugging modules together in an interpreted script file for ease of development and experimentation. Provisions have been made to support the execution of time- and event-changing network models and devices can be connected through several distinct communication channels (e.g. regulation, transcription). Communication channel characteristics (e.g. a model of molecular diffusion) can be customised and also selected from an interpreted script file.

Communications channels are modelled as *dynamic directed graphs*. Message passing over these channels can be modelled *stochastically*. Devices are able to consult their program *rules* to determine which action to take next either when triggered by an external message (e.g. protein), or caused by an internal condition (e.g. decay of an mRNA molecule). Further detail of modularity and efficiency provisions can be found in Section A.2.

3.7.2 Network Construction

This section describes how a network of nodes, connections and their properties are constructed, given a simulation script file from the generator.

3.7.2.1 Nodes

A candidate tape description takes the form of one or more lists of symbols that map to different types of BioSim devices. For example, in Figure 3.7.2.1, a type of node, O_{R2}, is defined. This node type is an operator with various effects on nearby promoters when CI or Cro is attached to it. Based on the notation presented in Section 1.7.2, the candidate tape is composed of a series of promoters, operators, encodings, terminators, oligos for recognition sequences and information about the plasmid backbone. For each symbol in the candidate tape description, a node of the specified type is created along with its configuration parameters. These parameters are described in Section 3.7.3. A unique node encompassing all other nodes that represents the host cell, holding proteins and mRNA, is also created with configuration for decay and translation rates.

```
<oR2>   $ Operator(oR2,...
          cI;...
          30%<attract>;...
          2%<decay>;...
          pRM;2500%<effect>|
          cro;...
          30%<attract>;...
          2%<decay>;...
          pR;9%<effect>)
```

Figure 3.7.2.1: An example definition of a operator node type. The operator is called O_{R2}, and the proteins CI and Cro have the same affinity value for it (0.3). When a CI protein attaches to an O_{R2} site which is near a P_{RM} promoter, the promoter will be sharply upregulated ($\times$ 25). Also, when a Cro protein attaches to an O_{R2} site which is near a P_R promoter, the promoter will be substantially downregulated (0.09). When either protein is attached to this type of site, there is a 0.02 probability, per time unit, that the protein will degrade.

3.7.2.2 Graphs

Nodes are connected together sequentially as they appear in the candidate tape description in bi-directional *helix* graphs that represent physical connectivity between the sites. If a tape is a plasmid then the helix graph forms a loop of sites. No messages are sent directly over a *helix* graph. Instead, they influence possible transcription routes, and regulatory effects through site proximity.

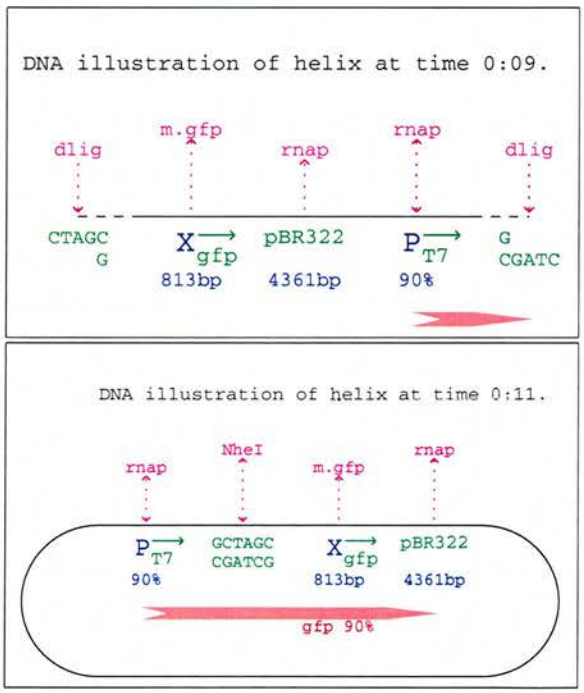


Figure 3.7.2.2a: Illustrations of the same piece of DNA in linear form and plasmid form. The linear piece of DNA (above) has two *Nhe*I cohesive ends that are compatible. It also holds X_{gfp}, a *gfp* gene, and P_{T7}, a promoter, but as the DNA is severed, transcription cannot take place, as RNA polymerase will fall off the end of the tape after attaching to the promoter. When a ligase has joined and circularised this piece of DNA into a plasmid (below), the promoter is now upstream of the *gfp* gene, thus allowing transcription to occur.

The *helix* graphs may be broken and rejoined dynamically, during the course of a simulation, by DNA-arranging enzymes and this, in turn, can alter regulatory influences and transcription routes (Figure 3.7.2.2a).

The *regulatory* graph is formed by connecting operator sites to nearby promoter sites (on an unbroken length of helix) that have the same orientation. These operator sites can broadcast regulatory messages to any nearby promoters each time there is a potential change in their regulatory effect, such as when a CI protein attaches to an O_{R2} site (Figure 3.7.2.2b). The promoters keep track of regulatory messages and update their affinity for RNA polymerase accordingly. Messages sent over the *regulatory* graph represent inferred behavioural state transitions resulting from stochastic events that have already occurred, that is molecular attachments and DNA rearrangement. Consequently, regulatory messages are transmitted instantly and reliably.

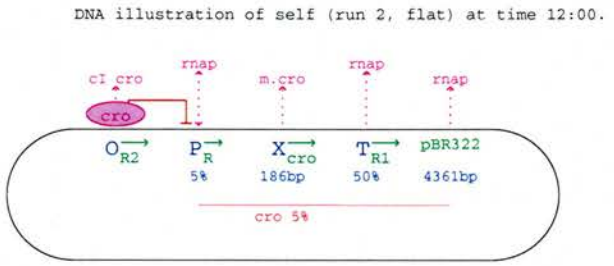


Figure 3.7.2.2b: An illustration of a regulatory message being sent from an operator site to a promoter. When the Cro molecule attaches to the O_{R2} operator, a regulatory message is transmitted from O_{R2} to the nearby promoter, P_R. Consequently, the P_R site reduces its affinity for RNA polymerase.

The *transcription* graph is formed in a number of stages. Each promoter is used as a starting point for transcription in turn, with its orientation dictating the direction that the *helix* graph will be traversed, one site at a time. If an encoding or terminator node, of the same orientation, is encountered, then an edge is formed from the previous connected node, or starting point, to this node. This allows encodings to receive RNA polymerases for transcription and allows terminators to regulate the RNA polymerases passing onward through them.

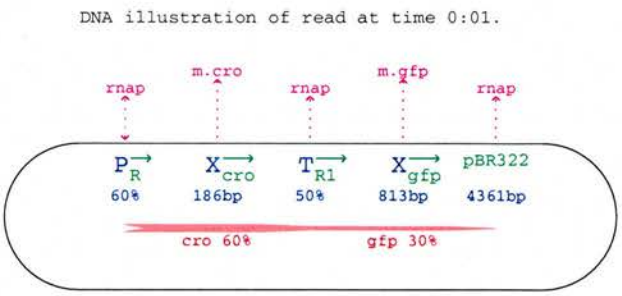


Figure 3.7.2.2c: An illustration of RNA polymerase reading through a terminator. The protein encodings, X_{cro} and X_{gfp} are both downstream of the P_R promoter. However, owing to the terminator site, T_{R1}, which blocks half of the RNA polymerases from passing over it, the anticipated average transcription rate of the *cro* gene is twice as high as that of *gfp*.

Edges also have distance attributes encoded which are determined from the numbers of base pairs of the sites. This allows time delays of an RNA polymerase's movement to be simulated as it moves along the DNA. If an oligo that has broken ends, a plasmid encoding or a protein encoding in the opposite direction is encountered, then transcription is considered to be effectively terminated by these sites. If another promoter with the same orientation is encountered, then we connect the graph to that promoter. This allows RNA polymerase to read over promoters after they have already attached at a different site, upstream (Figure 3.7.2.2c).

The *rna* graph represents the movement of proteins and mRNA molecules between the cell and potential accepting sites on the simulated DNA. Molecules are diffused stochastically over the *rna* graph to sites on the DNA. Depending on the type and quantity of the molecules in the vicinity of a site, and that site's current state (e.g. an upregulated promoter receiving RNA polymerase) a molecule may attach to a site with a calculated probability. Molecules which fail to attach are returned to the cell.

Various connections are made on the *rna* graph between the cell and DNA sites for the potential attachment (to site) and rejection (from site) of proteins. Connections from encoding nodes to the cell pass mRNA templates which are produced during transcription. Regulatory proteins pass between the cell and operator sites. DNA-arranging enzymes from the cell can attempt to attach to oligo sites and may be returned to the cell. RNA polymerase molecules from the cell may attach to promoters and may also be rejected by promoters, terminators and plasmid encodings. Figure 3.7.2.2d illustrates the different types of *rna* graph connection with an example tape.

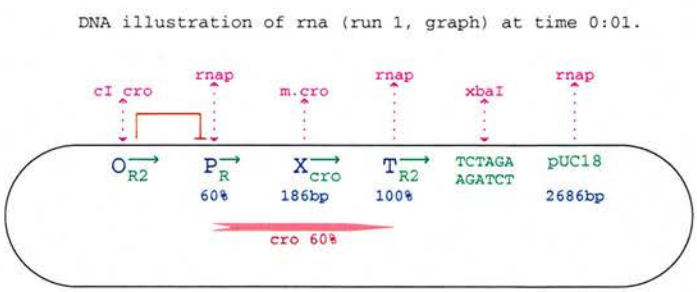


Figure 3.7.2.2d: An illustration of the types of *rna* graph connection. The operator, O_{R2}, can receive and reject Cro and CI proteins. The promoter, P_R, can receive and reject RNA polymerase (rnap). The protein encoding, X_{cro}, can produce mRNA Cro transcripts. The terminator, T_{R2}, and the plasmid, *pUC18*, can return RNA polymerase to the cell. Finally, the sequence 5'-TCTAGA-3' is recognised by the enzyme, *XbaI*, which can attach and detach to and from this site.

The *assemble* graph is used to send messages that describe changes which have occurred that modify the *helix* graph, and therefore potentially affect the consistency of the other types of graph. These messages are sent when oligos are severed and rejoined. Messages are transmitted instantly and reliably as they represent a completed state change (of the physical structure of the simulated DNA) which is a direct consequence of previous, stochastic interactions.

3.7.3 Node Operation

Next, the operation of each type of node is described in terms of its message passing over each of the graphs described, processing and state affecting its behaviour. Where a node is not shown to handle a specific type of protein (message), it is assumed that such proteins will always be rejected and returned to the cell node. Similarly, proteins that have an affinity for, but do not successfully attach to a site are also assumed to be rejected. Figure 3.7.3 summarises the graphs on which each type of node passes messages.

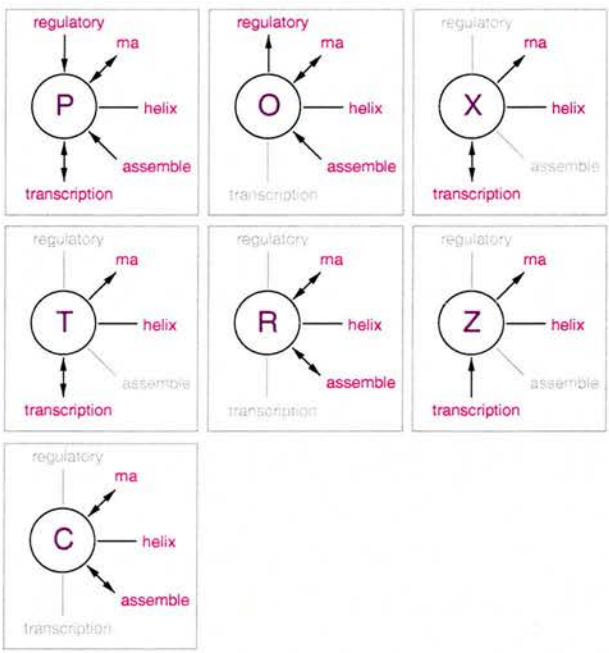


Figure 3.7.3: The simulation graphs on which each type of node communicates: (P) promoter, (O) operator, (X) encoding, (T) terminator, (R) oligo, (Z) plasmid, (C) cell. Arrows indicate the direction of message-passing, and shaded-out graphs are unused. Transcription messages may pass through promoters, encodings and terminators, and will be effectively stopped by plasmids. Proteins may be transfered, over the *rna* graph, between a cell and promoters, operators and oligos. Terminators and plasmids may re-route an RNA polymerase from a *transcription* to *rna* graph. All nodes are located on a double-helix. Oligos (when manipulated by enzymes) and cells may both trigger an arrangement event. These events can affect the state of all nodes except plasmids, terminators and encodings. Operators may send *regulatory* messages to promoters.

3.7.3.1 Promoter

A promoter is configured with a base affinity for RNA polymerase that determines the probability of transcription beginning in the absence of regulatory effects. For example, the viral P_{T7} promoter is defined in Figure 3.7.3.1.

```
> promoter pT7 affinity(0.9)
taatacgactcactatagggaga
```

Figure 3.7.3.1: The configuration of an example promoter site, P_{T7}. An RNA polymerase molecule which comes into contact with an unoccupied P_{T7} promoter will attach with a probability of 90%.

A promoter may also record a number of dynamic regulatory effects given to it by nearby operator sites, during the course of a simulation. These effects can alter the promoter's current affinity for RNA polymerase. Each RNA polymerase message received by a promoter attaches to it with a probability determined by the promoter's current affinity value. Only one RNA polymerase molecule can attach to a promoter per unit time. After a DNA tape has been rearranged, it is possible that some of the regulatory effects recorded by a promoter are invalid. Consequently, when a promoter is notified of a DNA rearrangement it clears its regulation records (and will be sent new regulatory messages from any nearby operators).

3.7.3.2 Operator

Operator sites are configured with a list of their regulatory effects on nearby promoter sites in addition to their nucleotide sequence. Each effect can cause the up- or down-regulation of a specified promoter site if a protein is attached to the operator site. The affinities of proteins for an operator and the decay rate of proteins when attached are also specified. For example, the bacteriophage λ operator, O_{R1}, is defined in Figure 3.7.3.2.

```
> operator oR1
> down(cI, 0.7, 0.02, pR, 0.98)
> down(cro, 0.1, 0.02, pR, 0.90)
atgcggccgggcc
```

Figure 3.7.3.2: The configuration of an example operator site, O_{R1}. A Cro protein which collides with an unoccupied O_{R1} site will attach with a probability of 10%, seven times less likely than a CI protein. When a Cro protein is attached to an O_{R1} site, its decay rate is reduced to 2% per unit time. Any nearby P_R promoter will be downregulated by 90% when Cro is attached.

When a protein message is received by an operator over the *rna* graph, if there is not already a protein attached, then it will attach with a probability determined by the sets of regulatory effects, if present within them.

At each time point, an operator with an attached protein determines whether or not it decays with a probability given by the corresponding regulatory effect field. An *RNAse* message, received over the *arrange* graph will also cause the decay of an attached protein. Whenever a protein attaches, detaches or decays an operator will broadcast its new regulatory effects to promoters nearby. A broadcast will also occur when DNA is rearranged, as the promoters that are nearby may have changed.

3.7.3.3 Encoding

Each type of protein is configured with a decay rate, per simulated unit time, for proteins which are not attached to DNA. The nucleotide base sequence encoding, from which the protein's mRNA can be generated is also specified. For example, the protein encoding, X_{cro}, is defined in Figure 3.7.3.3.

```
> protein cro decay(0.03)
atgtacaaga aagatgttat cgaccacttc ggaacccagc
gtgcagtagc taaggcttta ggcattagcg atgcagcggt
ctctcagtgg aaggaagtta tcccagagaa agacgcatac
cgattagaga tcgttacagc tggcgccctg aagtaccaag
aaaacgctta tcgccaagcg gcgtaa
```

Figure 3.7.3.3: The configuration of an example protein encoding site, X_{cro}. At each discrete time point, each Cro protein will decay with a probability of 3%.

Encoding nodes also record the time elapsed since the last RNA polymerase successfully began transcription so that the transcription rate can be limited. When a RNA polymerase molecule is received by an encoding node on the *transcription* graph, a check is made to verify whether there is space for it to continue transcription based on the time since the last transcription began. The RNA polymerase continues to the next site connected on the *transcription* graph, if there is space, and an mRNA message corresponding to the configured protein is transmitted over the *rna* graph. Both the edges of these graphs are configured with time delays based on the number of codons that the encoding holds. RNA polymerase molecules that are unable to attach are returned to the cell.

3.7.3.4 Terminator

Each type of terminator site is configured with its effectiveness of blocking the host cell's RNA polymerase, and its nucleotide sequence. For example, the T_{T7} terminator is defined in Figure 3.7.3.4.

```
> terminator tT7 effective(0.9)
caaaaaaccc ctcaagaccc gtttagaggc
cccaaggggt tatgctag
```

Figure 3.7.3.4: The configuration of an example terminator site, T_{T7}. A transcribing RNA polymerase molecule which encounters a T_{T7} terminator site, oriented in the direction of its movement, will be detached from the DNA with a probability of 90%.

When a RNA polymerase molecule is received on the *transcription* graph, that molecule is redirected back to the cell over the *rna* graph with a probability determined by the terminator's effectiveness. The molecule continues onward over the *transcription* graph if it is not redirected.

3.7.3.5 Oligo

Oligos are configured with an initial nucleotide base sequence, e.g. $R_{CCTAAGC}$, that is recognised by restriction enzymes that may be expressed during a simulation. They are represented at a lower level of abstraction, to allow the modelling of DNA arrangement (alongside gene regulation), and hence require more detail to describe.

A database is created for each simulation that holds configurable information allowing the calculation of the probability of enzymes modifying oligo sites and their effects, and the probabilities and effects of enzyme-mediated interactions between two oligo sites. As enzyme-sequence pattern-matching can be computationally expensive, a cache of results is maintained. Proteins may be conferred with digestive properties. A recognition pattern, restriction points and an indication of the enzyme's digestive effectiveness[9] relative to other enzymes are specified. For example, the digestive properties of *Bpu*10I are specified in Figure 3.7.3.5.

[9]Effectiveness is measured as the number of units of an enzyme required for an overnight incubation of 1μg DNA (Fermentas figures were used). A restriction endonuclease unit is defined as the amount of enzyme required to hydrolyse 1μg of bacteriophage λ DNA in 60 minutes in a reaction volume of 50$\mu\ell$.

```
> restrict bpu10I CCTNAGC +2 +5 0.2
```

Figure 3.7.3.5: The configuration of an example enzyme, *Bpu*10I. From the start of its recognition pattern, this enzyme nicks the upper strand of DNA two bases along, and the lower strand, five bases along. It requires 0.2 units for overnight digestion.

An oligo records its current nucleotide base sequence, including breaks and cohesive ends, and whether a protein has successfully attached to part of its sequence. When a protein approaches an oligo site, the database is consulted to determine the probability of that protein attaching and altering the oligo's sequence. When an alteration occurs, the oligo sends a message over the *arrangement* graph so that other graphs can be dynamically modified, if affected. Similarly, an oligo's sequence, or sequences, can be modified as a result of an interaction with another oligo. These interactions occur when a DNA-joining enzyme is in the proximity of a pair of oligo sites that are also next to each other. The relative likelihood of a linearised plasmid's ends being together versus two separate DNA fragments is also accounted for[10]. Each time an arrangement occurs, an entry is added to a datafile that describes the current layouts of the tapes so that the generator can evaluate rules in the criteria that require information of the tape layouts.

3.7.3.6 Plasmid

Plasmids are configured with a nucleotide base sequence. For example, the start of the definition of plasmid, *pUC18*, is shown in Figure 3.7.3.6.

```
> plasmid pUC18
tcgcgcgttt cggtgatgac ggtgaaa...
```

Figure 3.7.3.6: The configuration of an example plasmid, *pUC18*.

The plasmid node serves two purposes: Transcription is considered to be effectively terminated when passing through a plasmid encoding, and it also allows for rules in the criteria that require a plasmid to be part of a tape for its propagation between cell generations.

[10]The `circularise` command allows the relative likelihood of two compatible cohesive ends of the same strand of DNA to anneal, against that of two separate strands of DNA with mutually cohesive ends annealing (Section A.3.2).

3.7.3.7 Cell

The single cell node is configured with an mRNA decay rate, as a probability per unit time, per mRNA molecule; the mRNA translation rate is configured as a probability of starting a translation per unit time, per mRNA molecule and a series of proteins and their decay rates when not attached to a site. The cell node holds the current amounts of mRNA and proteins that are not attached to DNA. It also records a history of protein amounts during the course of the simulation which is used by the generator to assess conformance to most types of rule in the criteria.

The decay of mRNA and protein molecules is simulated stochastically over time with reference to their configured rates. Similarly, translation of mRNA into proteins is simulated stochastically. Protein molecules are broadcast by diffusion at each time point over the *rna* graph and may be received by sites. The cell node also accepts protein molecules which have been returned from sites and mRNA (from encoding sites) and adds them to its current amounts. When a cell node receives an *RNAse* message, all mRNA and protein amounts are reset to zero.

3.7.4 Candidate Evaluation

The generator receives a datafile of protein amounts and tape layouts over time for each trial's run that is simulated. Each of the rules in the criteria is checked for conformance against the data, and points are awarded according to each rule's importance[11]. A pool of the highest-scoring candidates is maintained during an experiment. A secondary score field for candidates is the size of the tape, either measured as the number of sites required to be joined together during wetware construction, or the length in nucleotide base pairs.

3.7.5 Candidate Optimisation

To allow for ease of analysis and manipulation of tape layouts by the generator, an operator-encoding layout was adopted that is regular in structure. Sometimes, redundant sites are included as a result of this regularity. Some redundant sites are considered because they become active after a DNA rearrangement. Some sites can be removed by optimisation after the search space has been analysed.

[11] Rules can be rated with an importance modifier on a five point scale from `quite important` to `extremely important` (Section A.1.4.6). By default, rules are rated with a score of `important`.

An optimisation approach was adopted that was similar to the strategy employed by François and Hakim in their candidate generation software [François and Hakim, 2004]. A candidate, or pool of candidates can be subjected to size optimisation, where the generator checks for redundant sites by removing each site in turn from a candidate, simulating the result and, if no deterioration occurs to its score, it is reinserted into the pool. This process is repeated until no further improvements can be made.

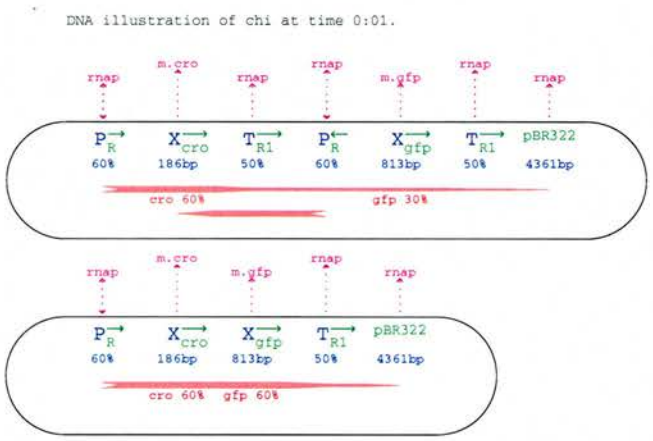


Figure 3.7.5: An example of a candidate DNA program tape's optimisation. The tape above holds a redundant promoter, P_R (pointing left) and a terminator, T_{R1} (3^{rd} from left) that will reduce the similarity of production between the two protein encodings. Removing both of these sites produces a more concise tape with more similar production levels from the protein encodings (below).

For example, the generator can produce a representation of *chimeric gene* – a synthetically modified natural gene – that is composed of one gene appended to another by inserting a non-functional promoter in between two encodings with a functional promoter up stream and weak terminators (Figure 3.7.5). By optimising the terminator away, the levels of mRNA production of each encoding will be more similar. Also, by optimising the non-functional promoter away, the level of mRNA production of the two encodings will still be tied to each other, with a smaller candidate.

Summary

This chapter has presented and justified the design of software that can generate representations of rearrangeable DNA programs. The next chapter will present experiments that used this software to accomplish three goals: verification by comparison of results with natural genetic programs and synthetic DNA programs, the investigation of the potential of a class of enzymes to support DNA reprogrammability, and to assess the potential scalability of DNA reprogramming.

Chapter 4

Simulation Experiments

The two aims of the simulation experiments were to investigate potential candidate DNA programs that could be used to demonstrate the concept of reprogrammable DNA and to show the potential scalability of self-reassembly using DNA-arranging enzymes. One of the simulation results, a two-colour, permanent DNA toggle, has also been implemented in wetware (Chapter 5).

4.1 Externally Settable Protein Toggle

The following experiments sought to find candidates able to act as a toggle switch [Gardener *et al.*, 2000], [Kramer, 2004] using protein concentrations, to hold the toggle's state in a mutually inhibitory network, rather than embedding this information permanently in DNA. The purpose of this was to allow a comparison of volatile (epigenetic) and non-volatile (genetic) memory approaches and to verify the generator's ability to find suitable gene-regulated examples.

4.1.1 Switch

The candidate criteria in Figure 4.1.1a specify a toggle switch which can be set with the protein CI and cleared with the protein Trp. The switch will fluoresce green when it is set. One of the simulation runs for the highest scoring candidate, a mutually inhibitory network, is shown in Figures 4.1.1b and 4.1.1c.

```
Trial:  first

Send 0.03 units of rnap from 0:00 to 16:00.
Send 0.03 units of cI from 0:00 to 0:10.

   Check gfp at 16:00 is more than 0.
   Check cI at 16:00 is more than 0.
   Check trp at 16:00 is approximately equal to 0.

Trial:  second

Send 0.03 units of rnap from 0:00 to 16:00.
Send 0.03 units of trp from 0:00 to 0:10.

   Check gfp at 16:00 is approximately equal to 0.
   Check cI at 16:00 is approximately equal to 0.
   Check trp at 16:00 is more than 0.
```

Figure 4.1.1a: A set of criteria for a toggle switch using proteins, CI and Trp. In the first trial, a short burst of CI is intended to cause the continued production of this protein and the suppression of *trp*. The second trial presents the opposite scenario, where a short burst of Trp maintains the expression of *trp* and suppresses the *cI* gene. The production of the fluorescent protein, GFP, is linked to the *cI* gene.

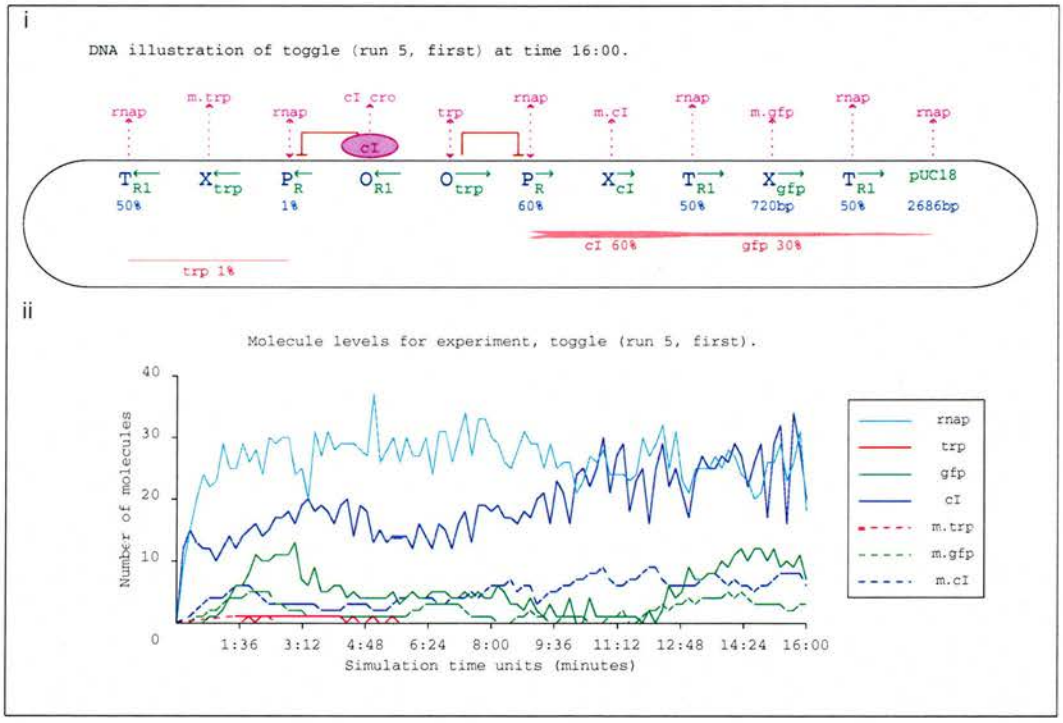


Figure 4.1.1b: Graphical simulation output of a generated toggle switch being set. (i) When the CI protein is introduced into the cell, it attaches to the O_{R1} operator and in doing so downregulates the left-facing P_R promoter, thus suppressing the production of Trp, as the *trp* gene is downstream of this promoter. CI is expressed from the right-facing P_R promoter, as is a smaller quantity of GFP, which reports the production of CI. (ii) When an external CI signal is sent, the amount of CI dominates the amount of Trp and there is a clear GFP signal indicating the switch's state.

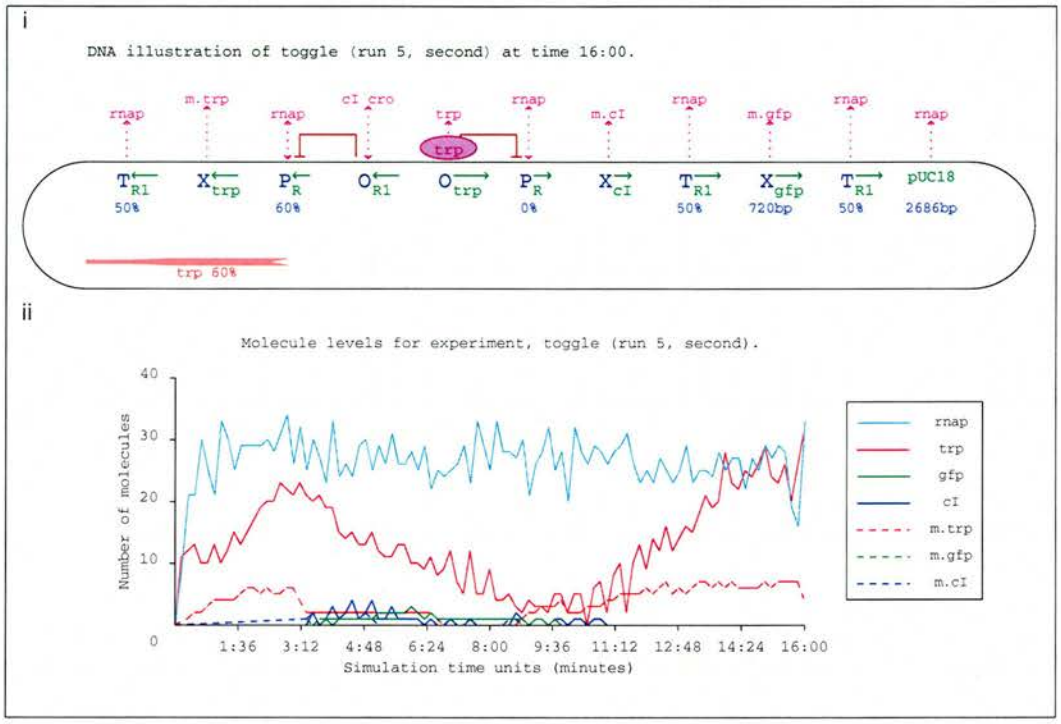


Figure 4.1.1c: Graphical simulation output of a generated toggle switch being cleared. (i) When Trp is introduced into the cell, it is able to bind to the O_{trp} operator which suppresses production of CI. Consequently, the *trp* gene is not suppressed by the *cI* gene. (ii) When an external Trp signal is sent, *trp* instead dominates *cI* and the GFP signal is very weak.

The state of the generated toggle switch can be changed, like a flip-flop device (Section 2.2), by exposing it to a complementary signal. For example, the state of a switch that has been cleared with a Trp signal can be subsequently set with a CI signal. The criteria in Figure 4.1.1a were extended by exposing switches that had previously been exposed to a Trp signal, to a CI signal and the results are shown in Figure 4.1.1d.

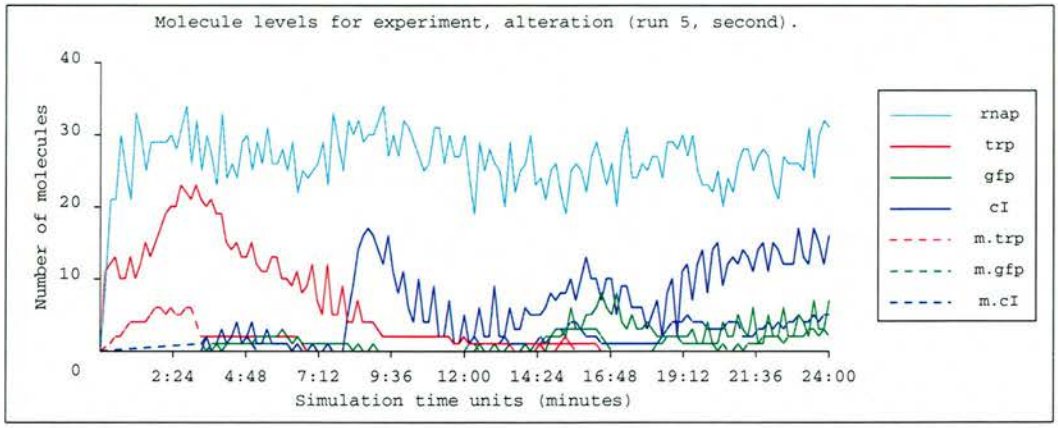


Figure 4.1.1d: Graphical simulation output of the alteration of a toggle switch's state. Initially, the toggle switch is exposed to a Trp signal which clears its state and suppresses the production of GFP. However, at time 8:00, the toggle switch is exposed to a CI signal, which sets its state and causes the production of GFP.

4.1.2 Two Colour Switch

The protein production rates of actual bacteria can vary widely depending on environmental conditions [Elowitz *et al.*, 2002]. Often, when measuring the level of a single output signal, such as a fluorescent protein in the previous simulation, experiments need to be repeated a number of times to gain statistical significance in addition to a range of experimental controls. Furthermore, in the case of a bacterium (unlike electronic gates), there is no entirely effective way to physically separate one part of a regulatory network from another as the "signals" are concentrations of molecules which react with each other stochastically. There will always be, at least, the potential for a low level of protein production from even a repressed site. For example, in the case of a toggle that fluoresces after it has been set, if a weak fluorescent signal is detected, this may be caused by a low level of protein synthesis by the cells, or by transcription leaking from a down regulated promoter.

Furthermore, nature often does not tend to produce optimally efficient binary (*full-on* and *full-off*) promoters in living cells, opting instead for more robust balancing mechanisms such as self-regulation (Section 2.3). Therefore, to allow for clearer results, the following experiment uses two signals for each state of a switch. A *relative* measurement of the signals provides a more reliable indication of the switch's state.

```
Trial:  first

Send 0.03 units of rnap from 0:00 to 8:00.
   Check gfp at 8:00 is much more than bfp at 8:00.
   Check gfp at 8:00 is more than 0.

Trial:  second

Send 0.05 units of IPTG from 0:30 to 4:00.
Send 0.03 units of rnap from 0:00 to 16:00.
   Check bfp at 8:00 is more than gfp at 8:00.
   Check bfp at 8:00 is more than 0.

   Check bfp at 16:00 is much more than gfp at 16:00.
   Check bfp at 16:00 is more than 0.
```

Figure 4.1.2a: A set of candidate criteria for a green-blue switch. If the switch receives no input signal then it will produce considerably more GFP than BFP. However, if the switch is set with an input signal of IPTG then it will instead produce much more BFP than GFP and will continue to do so after the input signal has ceased.

The candidate criteria in Figure 4.1.2a specify a switch that can be set with the modified sugar, *IPTG*. When the switch has not been set, it will glow more green than blue, i.e. the amount of *gfp* will exceed the amount of *bfp*. Once the switch has been set, it will glow more blue than green, even in the absence of the input signal. The highest scoring result found was of a network that triggered the production of an intermediatery protein, CI, in the presence of IPTG which *indirectly* upregulated a *lac* operator upstream of *cI*[1]. Once production of CI had started, it was able to upregulate a different operator (O_{R2}, P_{RM}) further upstream which also points towards the *cI* gene. Furthermore, *cI* is able to suppress *gfp* when CI proteins attach to the O_{R1} operator. Thus the production of CI is maintained even after IPTG has decayed as *cI* is able to sustain its own production. The mechanism found is similar to that in bacteriophage lambda which places it into its lysogenic phase [Herskowitz and Hagen, 1980]. One of the simulation runs of this candidate can be found in Figures 4.1.2b and 4.1.2c.

[1]Lac proteins are continually expressed by some host bacteria, such as *E. coli* JM109 and can bind to a *lac* operator site and cause downregulation. When IPTG binds to Lac proteins, the Lac proteins lose their ability to react effectively with *lac* operator sites. Therefore, IPTG indirectly upregulates *lac* operators in cells which continually express a *lac* gene.

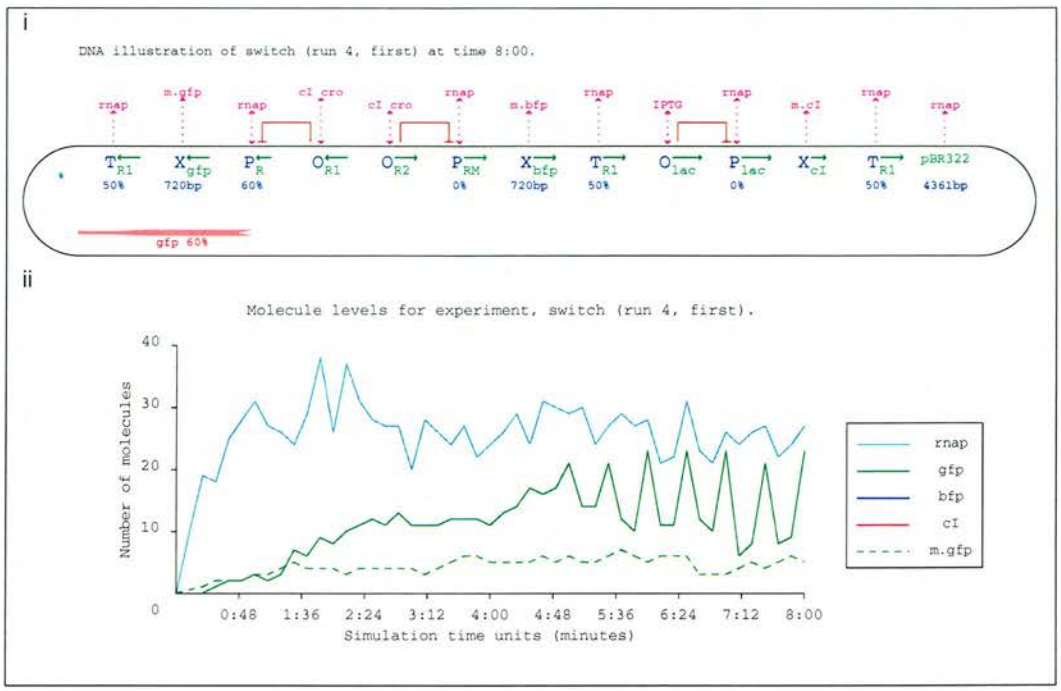


Figure 4.1.2b: Graphical simulation output of a green-blue switch without an external signal. (i) In the absence of an external signal, GFP mRNA is transcribed from the left-most P_R promoter. (ii) Without an IPTG signal, there is a clear GFP reading and no BFP reading. All molecules that are provided externally or have corresponding genes in the candidate are included in the key.

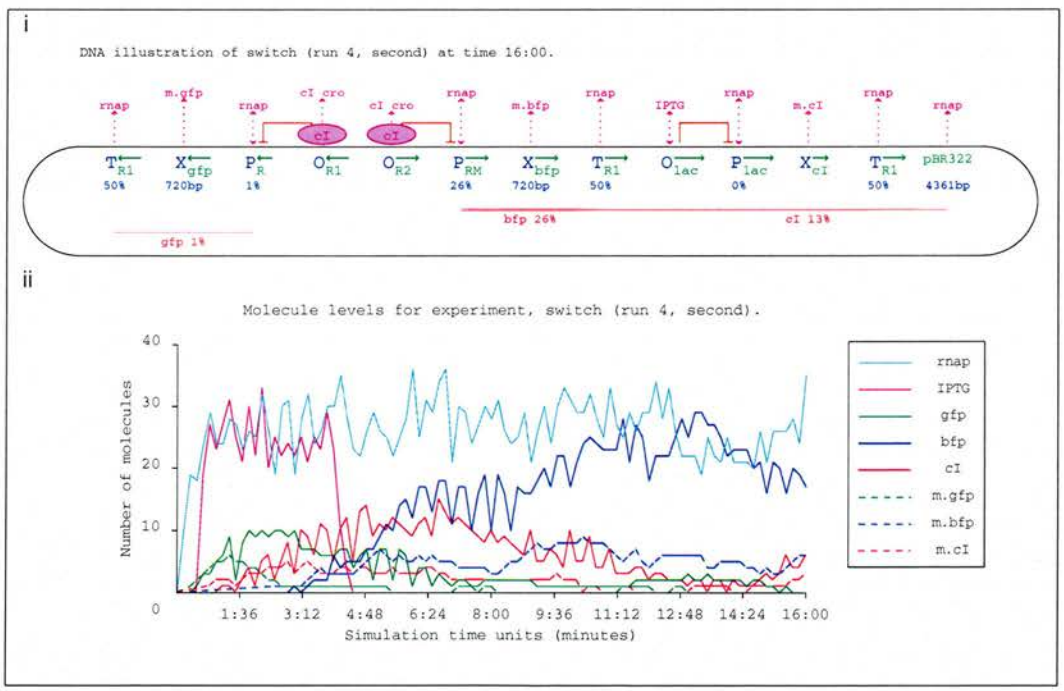


Figure 4.1.2c: Graphical simulation output of a green-blue switch with an external signal. (i) When an IPTG signal is sent this causes upregulation of promoter P_{lac} which triggers the production of CI. *cI* is able to maintain its own expression by upregulating the P_{RM} promoter when CI proteins attach to the O_{R2} operator. This also causes the production of BFP. The left-most promoter, P_R is downregulated by CI protein attachment, reducing the production of GFP. (ii) When IPTG is sent, CI is produced and maintained and the amount of GFP decreases and is then exceeded by the amount of BFP.

4.2 DNA Toggle

The switches presented in the previous example rely on the continuing production of decaying proteins by a cell to maintain their state. If the cell dies or is subjected to a counter signal, the original event of setting the switch can be lost. Furthermore, an extra burden is placed on the cell while it is producing the proteins required to maintain the switch's state. An analogy can be drawn between computer software on a hard-disk with DNA (non-volatile) and data in computer memory with proteins (volatile). The following set of experiments investigated how a number of categories of DNA restriction endonuclease, in conjunction with DNA ligase, may be used as part of a toggle switch that stores its state in DNA.

Restriction endonucleases were chosen over other DNA-arranging enzymes for the following reasons. Cohesive ends can be produced in a modular manner which aids scalability, i.e., different restriction sites can be digested to produce the same type of cohesive end. A large range of REs is available commercially for *in vitro* use, and many have been sequenced and can be expressed *in vivo* in *E. coli* [Roberts *et al.*, 2003].

Each of the enzymes chosen for the following experiments had been previously sequenced, cloned and was commercially available, to allow the potential for follow up wetware experiments. Only enzymes making modifications near to their recognition sequences, in a predictable (Type IIP and Type IIS) manner were investigated, otherwise a more complex simulation method would be required (Section 3.7). Five categories of restriction endonuclease were simulated to cover a number of possible types of recognition sequence, cohesive end and sequence ambiguity (Table 4.2a).

enzyme	sequence	pattern	cohesive	base *N*
*Asi*SI	GCG↑AT↓CGC	symmetric	symmetric	none
*Bbv*CI	CC↓TCA↑GC	asymmetric	symmetric	none
*Bpu*10I	CC↓TNA↑GC	asymmetric	symmetric	middle
*Bsm*I	GAATG↑CN↓	asymmetric	[a]symmetric	end
*Scr*FI	CC↓N↑GG	symmetric	symmetric	middle

Table 4.2a: The five Type II enzymes used in the DNA switch simulation experiments.

The candidate criteria for a DNA switch were based on the same requirements to which the protein switch was subjected (Section 4.1.2), with the additional condition that the switch should maintain its state, even if all proteins are removed. Also, for clarity, each of the five enzymes was sent directly, rather than being under regulatory control.

Preliminary experiments frequently produced linear fragments of DNA as the end result of a rearrangement. Such fragments come with two disadvantages: they cannot be replicated by a cell when it grows and the cell may also attack them. Consequently, rules were added to the criteria to emphasise the importance of being able to produce plasmid DNA after any rearrangements. An example set of criteria used for *AsiSI* can be found in Figure 4.2b.

```
Trial:  first

Send 0.03 units of rnap from 0:00 to 8:00.
   Check gfp at 8:00 is much more than bfp at 8:00.
   Check gfp at 8:00 is more than 0.

Trial:  second

Send 0.03 units of rnap from 0:00 to 8:00.
Send 0.05 units of dlig from 0:30 to 4:00.
Send 0.05 units of asiSI from 0:30 to 4:00.
   Check bfp at 8:00 is more than gfp at 8:00.
   Check bfp at 8:00 is more than 0.

Send reset at 8:30.

Send 0.03 units of rnap from 9:00 to 16:00.
   Check bfp at 16:00 is much more than gfp at 16:00.
   Check bfp at 16:00 is more than 0.
   Check dna at 16:00 with gfp is circular (very
      important).
   Check dna at 16:00 with bfp is circular (very
      important).
```

Figure 4.2b: The criteria for a switch that can be set using *Asi*SI with Dlig. When the switch has been set it will glow predominantly blue, otherwise it will glow predominantly green. The switch must be able to remember and report its state, even if all proteins are removed from the simulation (reset). Emphasis is placed on the need for the protein encodings to be located on a plasmid after any DNA rearrangements have taken place.

The generator found candidate tapes that used a number of different techniques to meet the criteria with varying success. The most successful candidates used a combination of techniques. The categories below discuss each in turn .

4.2.1 Knock Out

In this technique, the *gfp* gene, flanked with recognition sequences, was excised and the main plasmid and the gene fragment were both recircularised in a manner similar to a recombinase operation (Section 2.5.4). This separated the *gfp* gene from a promoter that was previously upstream of it. Although these knock outs were reversible for restriction endonucleases that recognise symmetric patterns, in general, the circularisation of a single piece of DNA is a much more likely event than the joining of two separate pieces. A variation on this method removed a promoter upstream of a *gfp* gene instead. An example encoding knock out using *Asi*SI is shown in Figure 4.2.1.

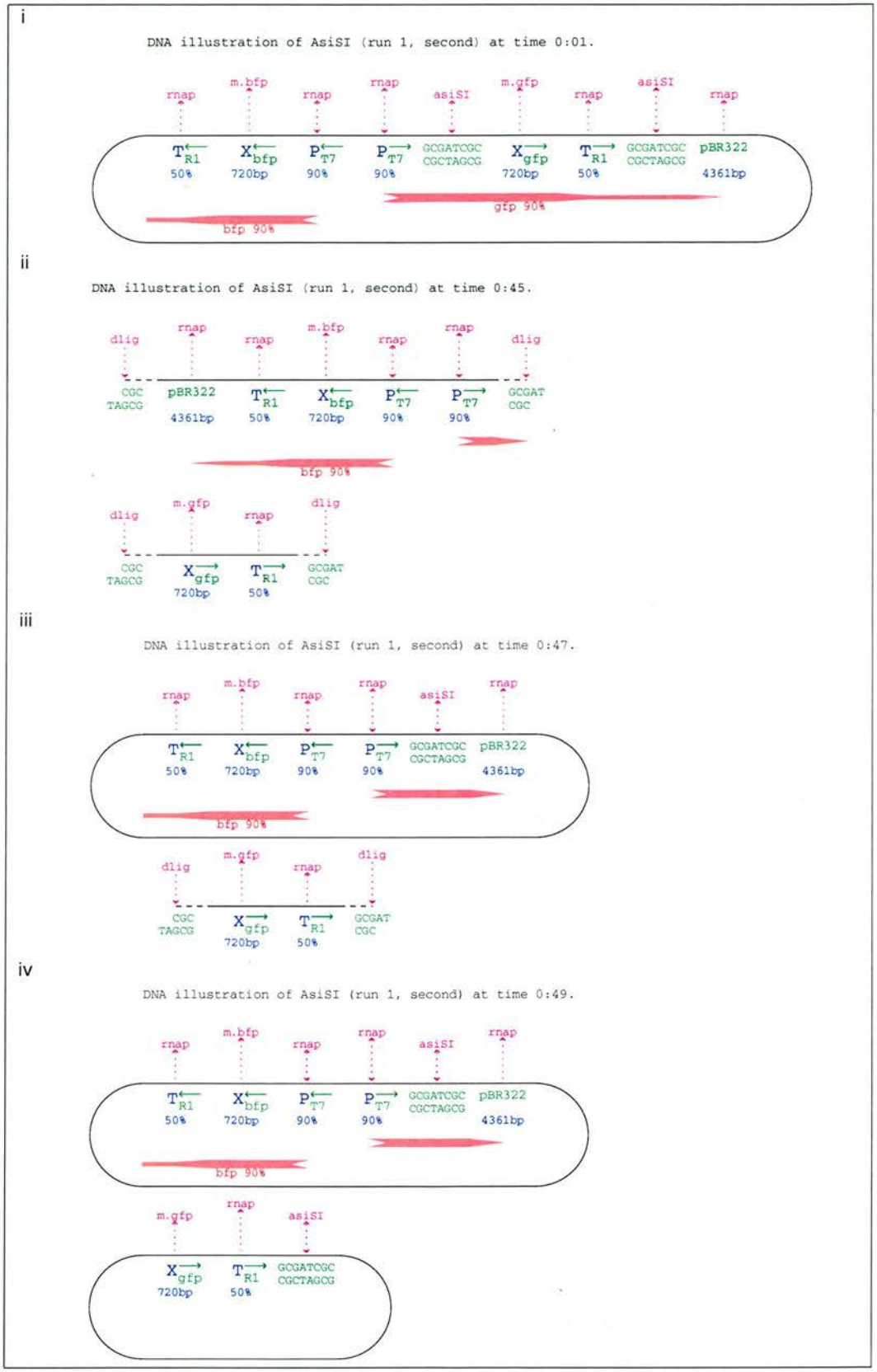


Figure 4.2.1: Graphical simulation output of an example *gfp* knock out using restriction enzyme, *Asi*SI. The figure description is continued overleaf.

Figure 4.2.1 continued: (i) Before modification of the candidate tape, GFP mRNA is transcribed from the right-most promoter, and BFP mRNA from the left-most promoter (both P_{T7}). Two identical recognition sequences for *AsiSI* flank the gene encoding, X_{gfp} and its terminator, T_{R1}. (ii) The tape, split into two parts, after it has been digested at both recognition sequences. Each part has two cohesive ends, all of which are compatible with each other. (iii) Ligase circularises the upper part of the tape, which now no longer includes the X_{gfp} gene encoding. (iv) The lower part of the tape holding the X_{gfp} gene encoding is circularised. The *gfp* gene cannot be expressed however, as there is now no promoter up-stream of its encoding. The new plasmids also contain recognition sequences recognised by *AsiSI* and are susceptible to further digestion.

4.2.2 Revealed Transcription

This method involved the removal of a section of DNA, flanked by recognition sequences, that blocked the transcription of a downstream gene encoding by a promoter upstream of the section. Removing this section of DNA "revealed" the promoter to the gene encoding, allowing transcription to take place. A variation of this technique removed a terminator with partial effectiveness, thus raising the level of read-through. An example of this variation, using enzyme *ScrFI*, is shown in Figure 4.2.2.

4.2.3 Rotation

In this method, a promoter pointing at a *gfp* gene is flanked by recognition sequences. The promoter is excised, and re-ligated in the opposite direction, now pointing at a *bfp* gene. This operation could only be performed when the cohesive end sequence was either of even size, or had an ambiguous base in the middle to allow the correct alignment of complimentary bases after rotation. Furthermore, rotation was very unlikely to occur when both of the flanking sequences were identical, as a knock out would be more likely owing to the strong preference for circularisation of a single strand of DNA. Using asymmetric recognition flanking sequences allowed irreversible rotations. An example of an irreversible rotation with non-identical flanking sequences using enzyme *Bpu10I* is shown in Figure 4.2.3.

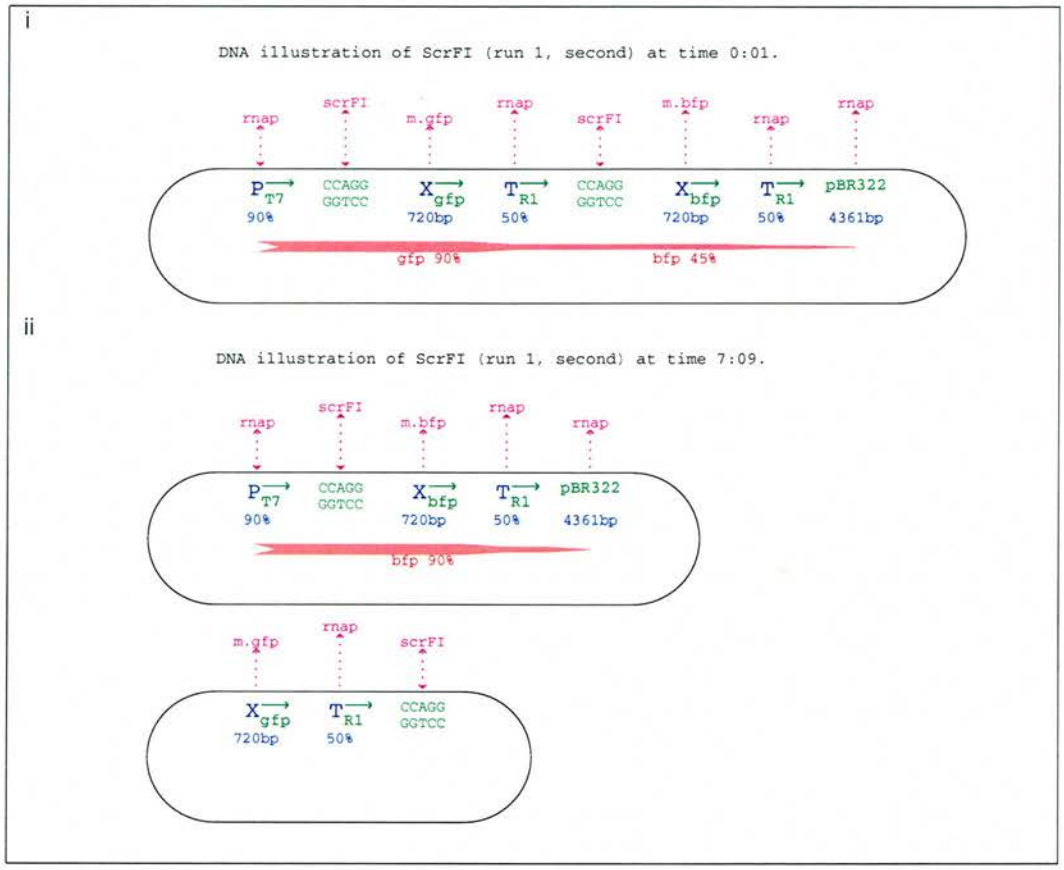


Figure 4.2.2: Graphical simulation output of a revealed transcription. (i) A single promoter, $\mathbf{P}_{T7}$ allows the transcription from gene encoding $\mathbf{X}_{gfp}$ and also, by reading through terminator, $\mathbf{T}_{R1}$, the transcription from $\mathbf{X}_{bfp}$. The $\mathbf{X}_{gfp}$ gene encoding and its terminator are flanked with recognition sequences for *Scr*FI. (ii) When the original tape is digested with *Scr*FI, the most likely ligations create two separate plasmids. One plasmid holds the excised $\mathbf{X}_{gfp}$ gene and its terminator and the other holds a $\mathbf{X}_{bfp}$ gene which is now down-stream of the promoter without a terminator in between to reduce transcription. The recognition sequences in the plasmids can be digested again with *Scr*FI.

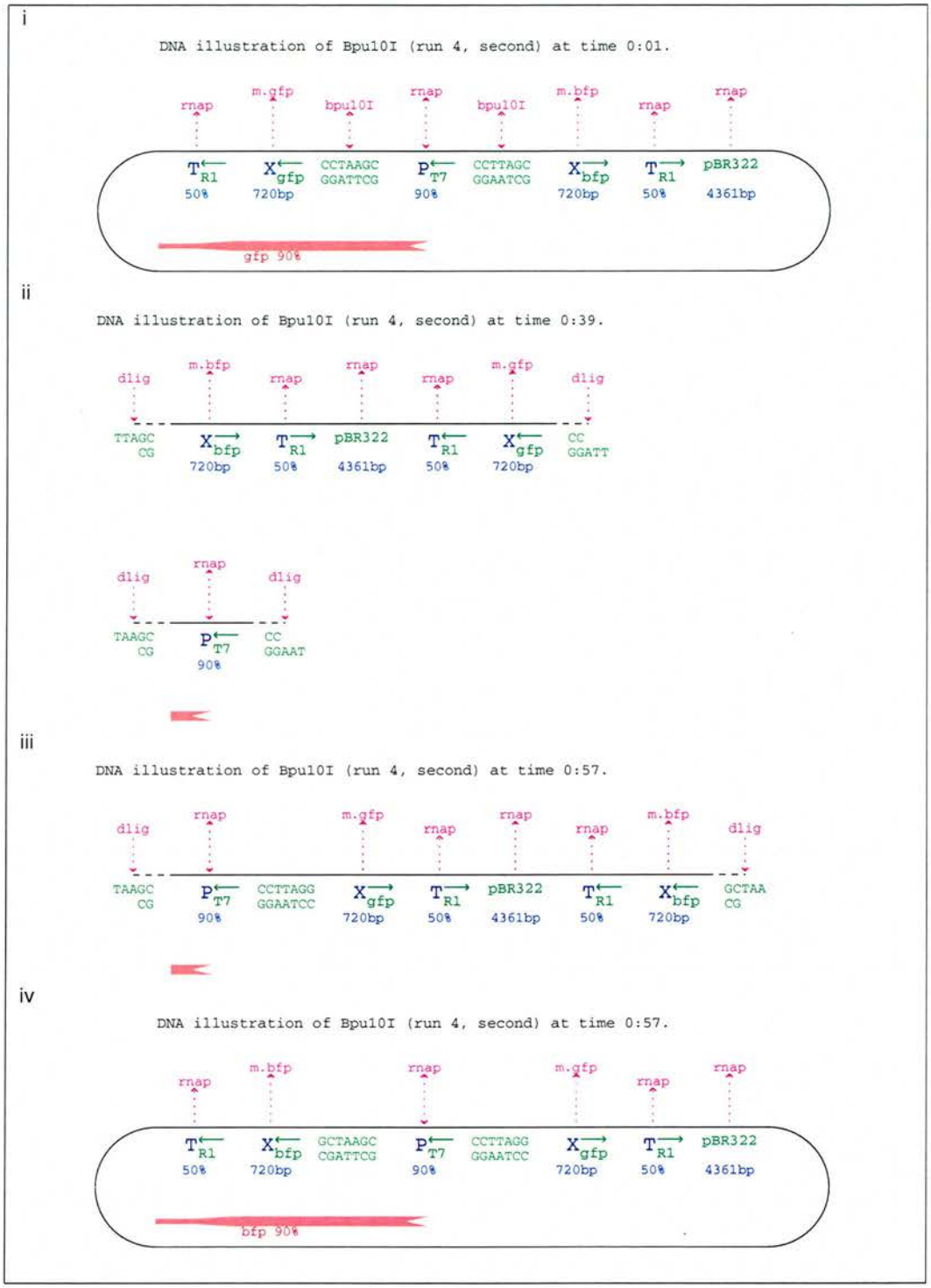


Figure 4.2.3: Graphical simulation output of a rotation operation using *Bpu*10I. The figure description is continued overleaf.

Figure 4.2.3 continued: (i) Before reassembly, the candidate tape expresses *gfp* from the central promoter, P_{T7} pointing towards the X_{gfp} gene. Up-stream of the promoter, is an inactive X_{bfp} gene. The promoter is flanked by two different sequences, both recognised by *Bpu*10I (the middle bases differ: A and T). (ii) Two pieces of DNA after both recognition sequences have been digested. The upper and lower parts cannot form plasmids separately owing to a base mismatch (middle bases). (iii) The two fragments ligate together with a different orientation. The new sequence shown to the right of the promoter cannot be digested with *Bpu*10I. (iv) Soon after the previous ligation, the DNA circularises. The P_{T7} promoter has been rotated and now points at the X_{gfp} gene. The second new sequence that has been produced can also not be digested by *Bpu*10I.

4.2.4 Enzyme Comparison

The top-ten scoring candidates for each enzyme were compared for their success in meeting the criteria. These results were influenced by the sequence recognition characteristics and by the strength of the bonds between complementary bases in cohesive ends which affect the probability of ligation. The range of scores and the number of times each technique was used, for candidates meeting at least two-thirds of the criteria, is described in Table 4.2.4.

enzyme	min	max	knock	reveal	rotate*	combined
asiSI	56	92	2	1	0	1
bpu10I	108	120	7	6	3	9
bbvCI	68	104	4	0	0	0
bsmI	76	92	5	2	0	2
scrFI	108	120	10	4	0	4

(*) Only candidate tapes that held a rotated section of DNA for the majority of the simulation time are counted. That is, temporary rotations which were undone are not counted.

Table 4.2.4: The range of the highest ten scores and the frequency that identified techniques were used for DNA switch candidates for a set of restriction endonucleases. The maximum score allowed by the criteria, 120, was obtained by two candidates. Some candidates used a combination of techniques, and these tended to score more highly.

There are other factors not taken into account by these experiments that would affect the plausibility of a DNA switch using restriction endonucleases. The length, in bases, of the recognition sequence determines the specificity (4^{length}). Higher specificity reduces the likelihood of recognising a host cell's DNA, which may need to be mutated or methylated to protect it from being digested (Section 2.5.2). *Bpu*10I is sixteen times more specific than the other high scoring enzyme, *Scr*FI. Also, the irreversible rotation operation used by some *Bpu*10I candidates produces a single plasmid after the DNA rearrangement with both fluorescent genes still located on it. This means that an inactive

gfp gene would still be inherited by subsequent generations of bacteria. This would be useful when more than one operation can be applied to a candidate, such as by having a second set of flanking sequences for a different enzyme that could reactivate the *gfp* gene.

4.3 Externally Settable DNA Switch

The previous experiment suggested that a DNA toggle could be realised using a single asymmetric restriction endonuclease with an ambiguous base in the middle of a symmetric cohesive end. The following experiment sought to place the mechanisms of that experiment under the control of an external signal, IPTG, using gene regulation. Such a switch could be used to make a permanent recording of the detection of a transient environmental signal. The candidate criteria can be found in Figure 4.3a.

```
Trial:  first

Send 0.03 units of rnap from 0:00 to 8:00.
Check gfp at 8:00 is much more than bfp at 8:00.
Check gfp at 8:00 is more than 0.

Trial:  second

Send 0.05 units of IPTG from 0:30 to 4:00.
Send 0.03 units of rnap from 0:00 to 8:00.
Check bfp at 8:00 is more than gfp at 8:00.
Check bfp at 8:00 is more than 0.

Send reset at 8:30.

Send 0.03 units of rnap from 9:00 to 16:00.
Check bfp at 16:00 is much more than gfp at 16:00.
Check bfp at 16:00 is more than 0.
Check dna at 16:00 with bfp is circular (very
important).
Check dna at 16:00 with gfp is circular (very
important).
```

Figure 4.3a: The criteria for a permanent blue-green switch that can be set externally using IPTG. When the switch has been set, using IPTG, it will glow predominantly blue, otherwise it will glow predominantly green. The switch must be able to remember and report its state, even if all proteins are removed from the simulation (reset). Emphasis is placed on the need for the protein encodings to be located on a plasmid after any DNA rearrangements have taken place.

One of the highest-scoring optimised examples found by the generator is shown in Figures 4.3b and 4.3c. The rotation operation performed in these experiments is irreversible (Figure 4.3d).

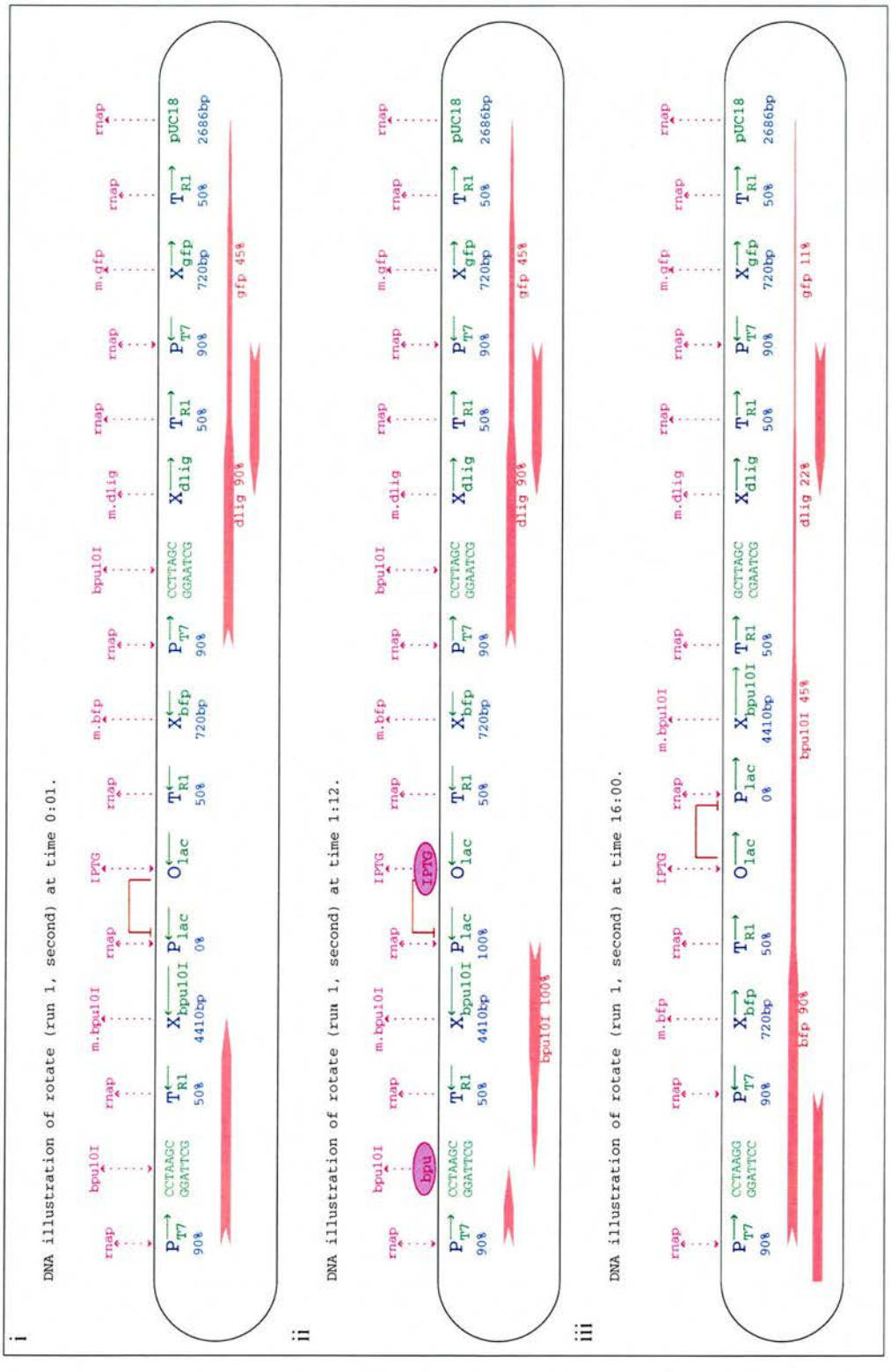


Figure 4.3b: Graphical simulation output of a candidate that enables a permanent green-blue switch to be set by an external IPTG signal. The figure text is continued overleaf.

Figure 4.3b continued: (i) By default, the candidate tape produces ligase and GFP. An initially redundant $\mathbf{P}_{T7}$ promoter is situated at the left of the tape and an inactive $\mathbf{X}_{bfp}$ gene is positioned close to the middle. Two sequences recognised by *Bpu*10I flank a collection of seven sites. (ii) When an IPTG signal is received, it upregulates the $\mathbf{P}_{lac}$ promoter causing the expression of *bpu10I*. (iii) The candidate tape after the seven site collection has been excised and then ligated back into the plasmid in the opposite direction. The originally redundant leftmost promoter now allows the transcription of $\mathbf{X}_{bfp}$. Smaller quantities of *Bpu*10I, ligase and GFP mRNA are transcribed. Ligations are generally less likely to occur than digestions (particularly in this case having three weak T and A bonds). Consequently, by producing ligase by default, this candidate tape had the advantage of having more ligase molecules available when the rotation operation occurred. As a result, this tape had more opportunity to rotate successfully.

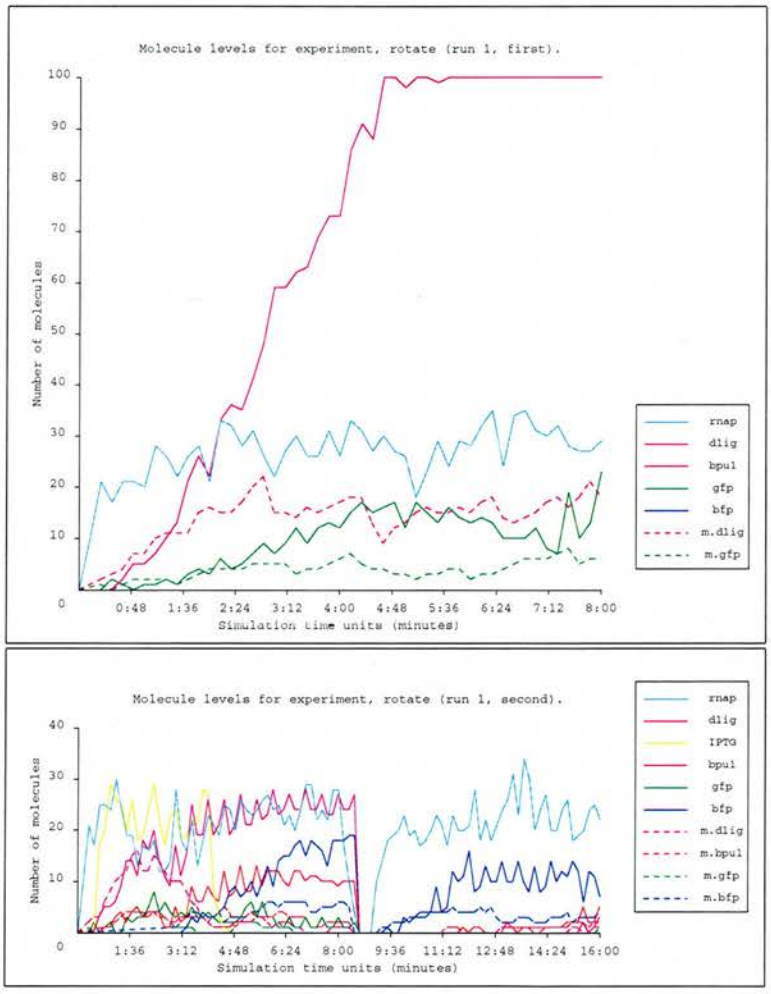


Figure 4.3c: Graphical protein amount plots of a candidate that enables a permanent green-blue switch to be set by an external IPTG signal. (upper) By default, there is a clear GFP signal and a large quantity of ligase produced. (lower) After an initial signal of IPTG, *Bpu*10I is synthesised, the tape is reassembled and a clear BFP signal exceeds the amount of GFP. When the new tape is simulated after removing all previous protein concentrations, the BFP signal still dominates the GFP signal and a small amount of *Bpu*10I and ligase is produced.

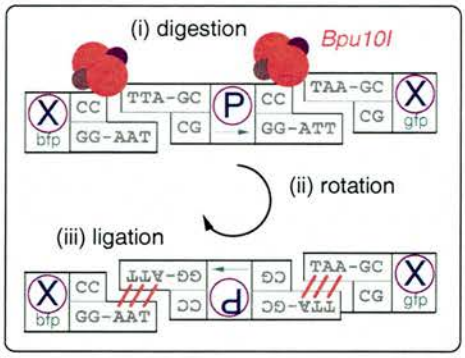


Figure 4.3d: Summary of the irreversible rotation operation encountered. (i) *Bpu*10I digests DNA at recognition sites flanking a promoter. (ii) The middle base of each recognition site is complementary to the other (*T* and *A*), thus allowing the excised section of DNA to attach in an inverted orientation. (iii) Once ligated into place, a rotated section of DNA is no longer recognised by *Bpu*10I, thus this particular manipulation of DNA is irreversible.

4.4 Combined Operations

There are significant numbers of sets of restriction endonucleases that, although recognising different sequences, can create compatible cohesive ends (Section 1.5.4). Other enzymes such as recombinases, lack this ability. For example, a set of restriction endonucleases, *Nhe*I, *Xba*I, *Avr*II and *Spe*I all produce the same compatible cohesive end, 3'-CTAG-5', but recognise different sequences. If two cohesive ends, produced by two different enzymes, from the four described, are ligated together, neither enzyme will be able to digest DNA at the resulting join. The simulation experiments in this section demonstrate the scalability advantage of being able to use combinations of DNA-modifying enzymes.

Firstly, a trivial case is presented, based on the excision of sites (Section 4.2.1), where four restriction endonucleases can be used to produce 16 (2^4) possible states. This example could also be accomplished using recombinases that are able to excise sections of DNA. Four genes are arranged sequentially on a tape, separated by strong terminators. Each gene's promoter is flanked by a pair of restriction sites. An input signal consists of sending a restriction endonuclease, which will excise the promoter of a gene, followed by a ligase to recircularise the tape. There is a small probability that an excised promoter will reinsert itself into the tape; however, given a suitable number of cells, majority assays can be taken. For each different restriction endonuclease sent, a gene will cease expression. Given that each gene is either on or off, one can record 2^4 possible states in DNA and have a host cell report the state through gene expression. Each promoter used on the tape can be regulated by an external signal, such as IPTG, to remove the protein synthesis burden placed on a cell while it is not required to report its state.

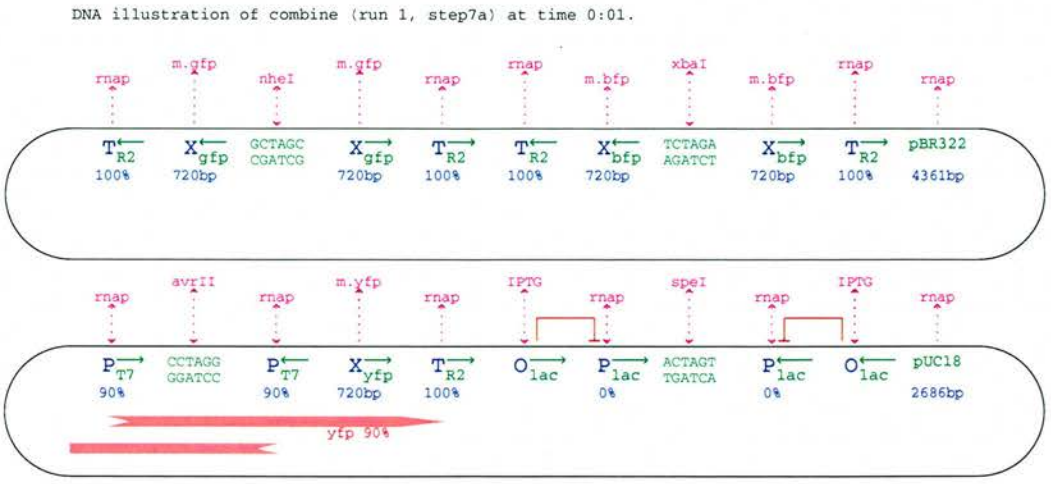


Figure 4.4a: Two library plasmids containing gene and operator sites. By sending combinations of enzymes with ligase, it is possible to rearrange the DNA of these plasmids to compose plasmids with new behaviours.

The main example of scalability presented exploits the use of combinations of enzymes, i.e., sending two or more restriction endonucleases, at a time, to cause a *cooperative* rearrangement effect on DNA. This example could also be used in conjunction with the trivial case, but would be more complicated to describe. Figure 4.4a shows two plasmids, one containing protein encodings and the other containing operators with a protein encoding in between. By sending combinations of restriction endonucleases, it is possible to manipulate these two plasmids to cause the construction of new genes. For example, by sending *Avr*II and *Nhe*I in the presence of ligase, two irreversible joins can be formed across the plasmids, allowing *gfp* to be transcribed from a T7 promoter. This kind of rearrangement is general purpose: the operators and encodings could be replaced with any kind of site. Restriction sites have operators and encodings mirrored around them so that an operator can always be positioned upstream of an encoding after a DNA rearrangement, no matter which particular way the broken plasmids happen to ligate[2]. Table 4.4b shows a set of combined restriction endonuclease signals sent to the two plasmids, and the simulated results, in terms of the output signals detected over ten simulation runs.

[2]Anti-sense transcripts (Section 1.5.3) will be formed from the operator oriented differently to an encoding, which will reduce the expression of the corresponding protein.

inputs		outputs			outputs with IPTG		
first	second	gfp	bfp	yfp	gfp	bfp	yfp
none	*none*	*0.0*	*0.0*	**1.0**	*0.0*	*0.0*	**1.0**
AvrII, SpeI	*none*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
AvrII, NheI	*none*	**0.9**	*0.0*	*0.0*	**0.9**	*0.0*	*0.0*
AvrII, XbaI	*none*	*0.0*	**0.9**	*0.0*	*0.0*	**0.9**	*0.0*
SpeI, NheI	*none*	*0.0*	*0.0*	**1.0**	**0.8**	*0.0*	**1.0**
SpeI, XbaI	*none*	*0.0*	*0.0*	**1.0**	*0.0*	**1.0**	**1.0**
AvrII, NheI	SpeI, XbaI	**0.8**	*0.0*	*0.0*	**0.8**	**0.8**	*0.0*
AvrII, XbaI	SpeI, NheI	*0.0*	**0.8**	*0.0*	**0.8**	**0.8**	*0.0*

Table 4.4b: Simulation runs with different expression directed by DNA-arranging input signals. Numbers refer to the proportion of runs in which a protein was clearly expressed.

It is also possible to obtain a state space of 36 using these four restriction endonucleases by combining the two examples presented so far in this section. Each restriction site on the plasmids (Figure 4.4a) can be duplicated at the same location, and have a gene inserted between the pair. Sending a matching enzyme will excise the gene. Whenever an enzyme is not required for a combined rearrangement (e.g. in row four of Table 4.4b *Spe*I and *Nhe*I are unused, allowing three additional combinations), it can also be used separately to excise a gene.

4.5 Count Down

The simulation platform does not support enzymes which are able to digest DNA a number of base pairs away from their recognition sequence owing to the complexity this would add (Section 3.7). However, using the reveal operation (Section 4.2.2), that was used by many of the generated candidates in earlier simulation experiments, one can imagine a DNA construct that would allow the counting of a number of discrete events, such as the number of times a cell divides using the scheme presented in Figure 4.5a.

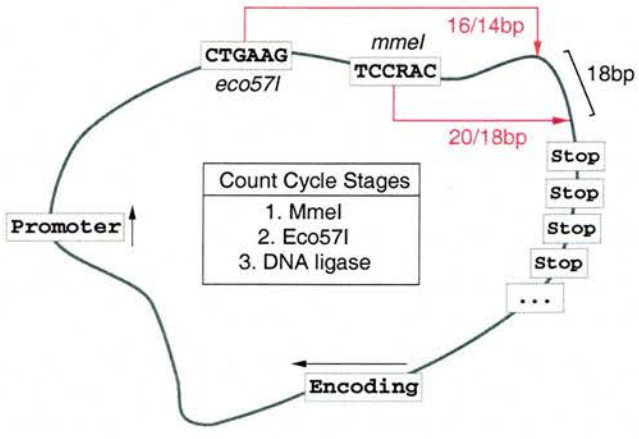


Figure 4.5a: An illustration of a potential DNA counter. At each discrete event that is to be counted, three signals are released in turn. The first two signals, *Mme*I and *Eco*57I, excise a terminator from the DNA and the third signal, a DNA ligase, recircularises the DNA without the excised site. When the last terminator has been excised, this reveals a gene to a promoter allowing its expression.

Two enzymes which digest beyond their recognition sequences are positioned so that one's restriction site sits between the other enzyme and its restriction site. A promoter points towards an encoding (such as for a fluorescent protein or apoptosis-causing protein) with a series of strong terminators in between. Consequently, the gene is not transcribed owing to the terminators' effects. At each time event, three signals are released in turn. Firstly, the right-most enzyme is transcribed causing a break in the DNA, then the left enzyme is transcribed causing a part of the DNA (one of the terminators) to be excised. After the restriction enzymes have been made to decay rapidly, either through being tagged, or by releasing a "clean-up" enzyme, ligase is expressed to recircularise the DNA. Thus, one terminator is excised for each set of signals. When all terminators have been excised, the encoding at the end of the terminator list can be expressed. This encoding could suppress the production of the restriction enzymes to prevent further digestion of the DNA. This scheme therefore illustrates a potential DNA-based counter.

Summary

This chapter has demonstrated the capabilities of the DNA program software and has used it to demonstrate potential DNA program arrangement mechanisms and their scalability, in terms of modularity. The next chapter will describe how one of the potential DNA programs, a two-colour DNA switch generated by the software, was implemented and tested in wetware to assess the biological plausibility of this example DNA reprogramming operation.

Chapter 5

Biological Experiments

This chapter describes how a two-colour, DNA switch with a rotatable mid-section, found in simulation experiments (Section 4.2.3) was implemented and tested in wetware. The aims of the experiments were to demonstrate: the plausibility of a restriction endonuclease-driven DNA switch (Section 5.3.1); that such a switch's state is stable across generations of bacteria (Section 5.3.7); that a switch's state can be clearly distinguished (Sections 5.3.4 and 5.3.6) and that once set, a switch's state is irreversibly stored in DNA, even if further subjected to the signal that caused the state change (Section 5.3.7).

An introduction to the standard laboratory techniques used in the biological experiments can be found in Section 1.6. The first section in this Chapter describes how the plasmids used in experiments were constructed (Section 5.1). The subsequent section describes how the rotation experiments were carried out (Section 5.2). The third section presents the results of the rotation experiments (Section 5.3). A description of how standard techniques were performed can be found in the Appendix (Section B.1).

5.1 Construction

The construction process fell into three main stages (Figure 5.1a).

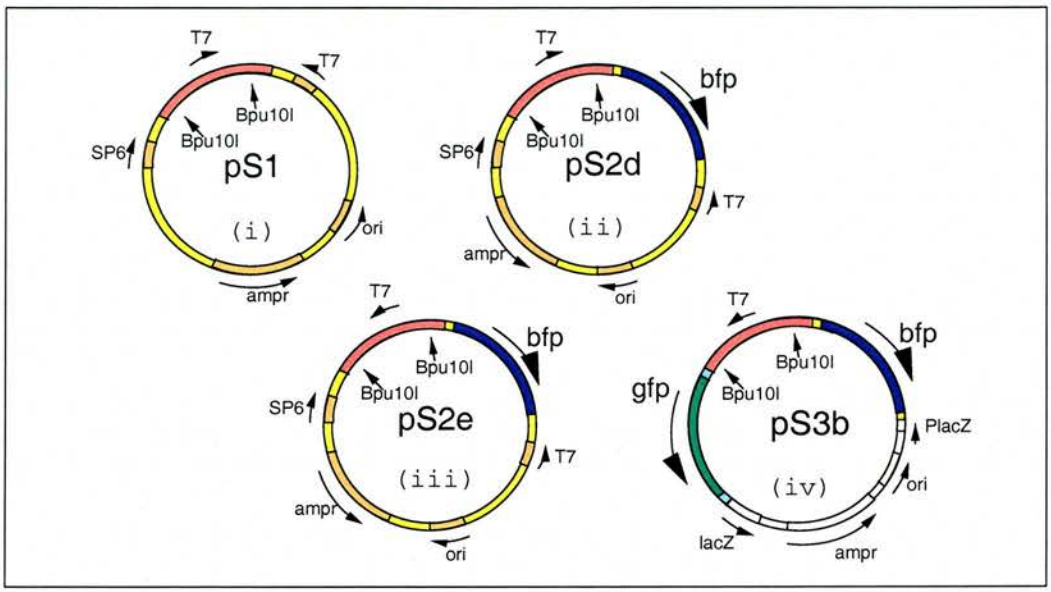


Figure 5.1a: Plasmids constructed during the three main wetware stages. During stage one, a T7 promoter is flanked by two *Bpu*10I restriction sites (i). Stage two extends stage one, by appending a fluorescent protein encoding oriented both downstream (ii) and in the opposite direction (iii) of this T7 promoter. In stage three, two different fluorescent protein encodings sit either side of a T7 promoter which is flanked by *Bpu*10I restriction sites (iv).

In the first stage, to be explained in detail below, a plasmid was constructed with a rotatable mid-section (Section 5.1.1) and was subjected to the restriction endonuclease, *Bpu*10I and T4 DNA ligase to induce the rotation of this mid-section. Plasmid DNA from before and after the rotation was sequenced and compared to determine the outcome (Section 5.2.1).

During the second stage, two plasmids were constructed based on the mid-section used in the previous stage: one with a blue fluorescent protein (*bfp*) gene downstream of a T7 promoter inside a rotatable mid-section, and the other with a *bfp* gene upstream of a mid-section (Section 5.1.2). *In vivo* gene expression assays were taken before and after both plasmids had been subjected to a rotation operation (Section 5.2.2).

The third stage extended the second stage by adding a green fluorescent protein (*gfp*) gene on the opposite side of the mid-section occupied by the *bfp* gene (Section 5.1.3). A T7 promoter inside the mid-section pointed towards the *gfp* gene before the rotation operation was applied. *In vivo* GFP and BFP expression was assayed before and after rotation (Sections 5.2.3, *"Reporting"* and *"Distinction"*). The stability of the switch in colonies, before and after rotation, was assessed by growing the bacteria from picked colonies in two overnight cycles, and then checking for any bacteria that had changed the state of their switch (Section 5.2.3, *"Stability"*). Finally, the irreversibility of the switch, once set, was also demonstrated *in vitro* by digesting DNA, from before and after rotation, with *Bpu*10I (Section 5.2.3, *"Irreversibility"*). A summary of the plasmids made during these three stages can be found in Table 5.1.

plasmid	description
pGEM-T	Cloning vector from Promega. Contains SP6 and T7 promoters.
pQBI-63	GFP vector from QBiogene. For bacterial expression from T7 promoter.
pQBI-T7-BFP	BFP vector from QBiogene. For bacterial expression from T7 promoter.
pS1	A rotatable promoter in a *pGEM-T* vector.
pS1-R	After *pS1* has been subjected to rotation, a section is expected to be inverted.
pS2a	A *bfp* gene in *pGEM-T* pointing the opposite way to its T7 promoter.
pS2b	A rotatable promoter in *pGEM-T*, in the opposite way to its T7 promoter.
pS2c	A rotatable promoter in *pGEM-T*, in the same direction as its T7 promoter.
pS2d	A *bfp* gene is pointed to by a rotatable promoter in a *pGEM-T* vector.
pS2e	A *bfp* gene is pointed away from by a rotatable promoter in *pGEM-T*.
pS2d-R	After *pS2d* is subjected to rotation, BFP should no longer be produced.
pS2e-R	After *pS2e* is subjected to rotation, BFP should start to be produced.
pS3a	Based on *pQBI-63*, a rotatable promoter sits between a *gfp* and *bfp* gene.
pS3b	A rotatable promoter in *pUC18* points at *gfp*, with *bfp* on the other side.
pS3b-R	After *pS3b* is subjected to rotation, should glow blue, not green under UV.
pUC18	Small, high copy number cloning vector.

Table 5.1: Summary of plasmids used during the construction stages and in experiments. Names with a suffix of 'R' have been subjected to a rotation operation. The number in the name of a plasmid beginning with 'pS' points to the stage during which it was constructed.

5.1.1 Stage 1

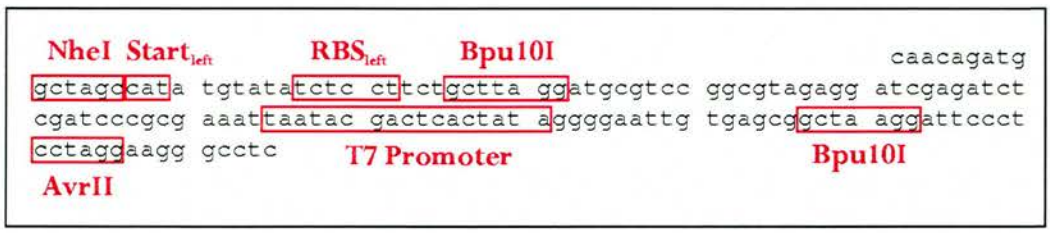


Figure 5.1.1: The transcription, rotation and construction components of the "longmer" oligo.

A 144bp double-stranded oligonucleotide, called the "longmer", was designed so that a bacteriophage T7 promoter was flanked by two different *Bpu*10I restriction sites, 5'-GCTTAGG-3' and 5'-GCTAAGG-3' (Figure 5.1.1). The middle base of the two restriction sites is complementary, so that when the 80bp mid-section between these sites is excised after digestion with *Bpu*10I, this excised mid-section is able to ligate back into the same place, in the opposite orientation. To aid the construction of latter *in vivo* stages, restriction sites were added for the appending of *gfp* and *bfp* genes. Restriction site, *Avr*II was added to allow the appending of a *bfp* gene. A restriction site *Nhe*I, a ribosome binding site and a start codon were added to allow the appending of a *gfp* gene. To reduce the chance of leaky translation caused by a transcript started from outside the "longmer", stop codons were placed in all three reading frames in both directions.

The ssDNA "longmer" sequence (Figure 5.1.1), which was synthesised by MWG, was amplified with primers, 5'-GAGGCCCTTCCTAGGAGGGAA-3' ($long_L$) and 5'-CAACAGATGGCTAGCCATATG-3' ($long_R$). The resulting DNA was gel extracted, ligated overnight into a *pGEM-T* vector and transfected into competent JM109 cells. Plasmid DNA extracted from these colonies by mini-prep (Section B.1.2.6) is referred to as *pS1*.

5.1.2 Stage 2

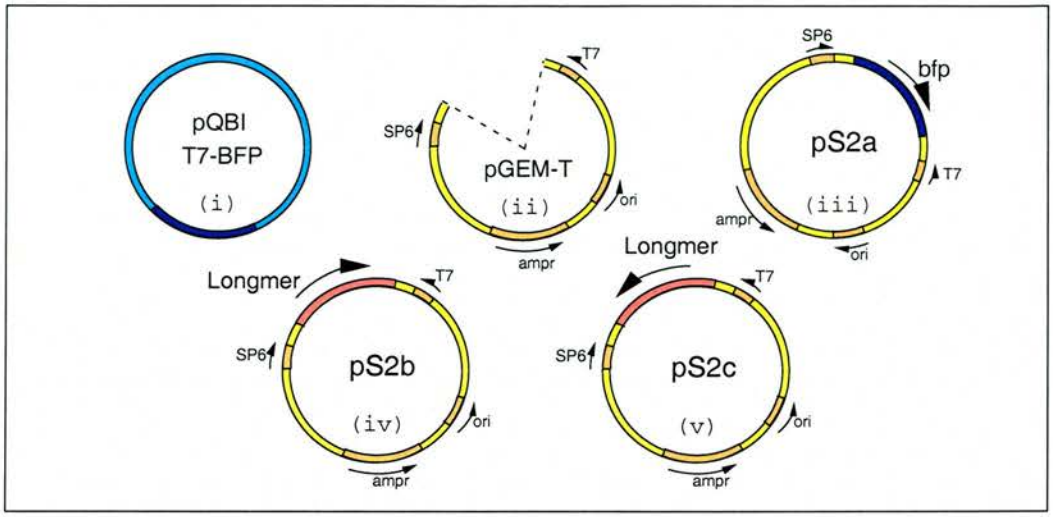


Figure 5.1.2a: Plasmids constructed during Stage 2*a*. (i) The *pQBI T7-BFP* bacterial BFP expression plasmid (QBiogene). *bfp* gene shown in dark blue. (ii) Linearised *pGEM-T* cloning and expression vector (Promega). (iii) *bfp* gene inserted into *pGEM-T* vector downstream of SP6 promoter (plasmid *pS2a*). (iv) Longmer oligonucleotide inserted into *pGEM-T* vector downstream of SP6 promoter (plasmid *pS2b*). (v) Longmer oligonucleotide inserted into *pGEM-T* vector downstream of T7 promoter (plasmid *pS2c*).

The *bfp* gene from *pQBI-T7-BFP* (Figure 5.1.2a *(i)*) was amplified using primers, 5'-CTTGGTTATGCCGGTACTGC-3' (bfp_L) and 5'-GGTGATGTCGGCGATATAGG-3' (bfp_R). The result was gel extracted, ligated into a *pGEM-T* cloning vector (Figure 5.1.2a *(iii)*) and transfected into JM109 cells (plasmid, *pS2a*). DNA was extracted by mini-prep after selection through blue/white screening (Section B.1.4.4). The orientations of the inserts were determined by digestion with *Xba*I to linearise the DNA upstream of the *bfp* gene, plus *Sac*II and *Sal*I, which digested either side of the vector cloning site. The insert pointing in the opposite direction to the *pGEM-T* T7 promoter, digested with *Sac*II, was gel extracted (Figure 5.1.2b *(ii)*).

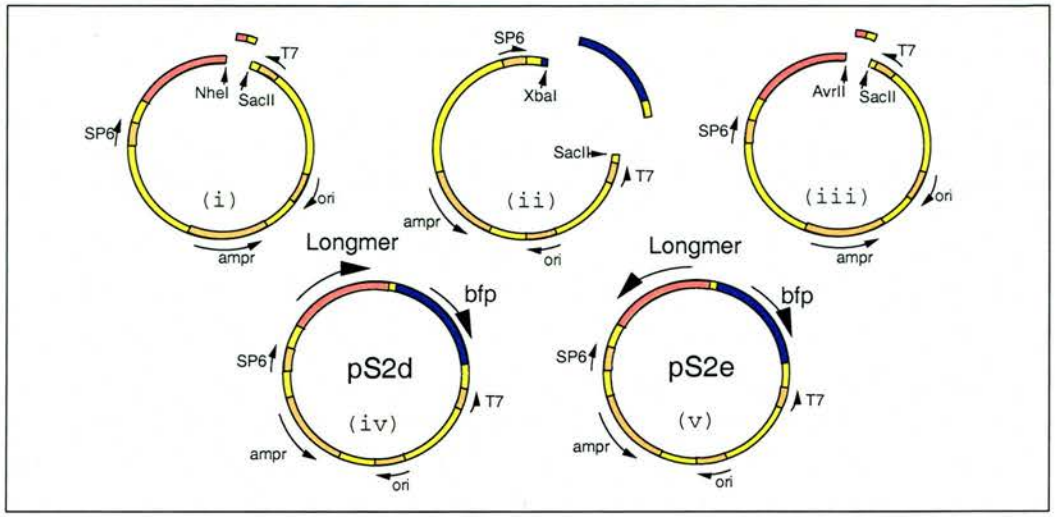


Figure 5.1.2b: Plasmids constructed during Stage 2*b*. (i) *pS2b* after digestion with *Nhe*I and *Sac*II. (ii) *pS2a* after digestion with *Xba*I and *Sac*II. (iii) *pS2c* after digestion with *Avr*II and *Sac*II. (iv) A Longmer oligonucleotide upstream of a *bfp* gene in a *pGEM-T* vector (plasmid *pS2d*). (v) A *bfp* gene and longmer oligonucleotide in opposing directions inside a *pGEM-T* vector (plasmid *pS2e*).

The dsDNA "longmer" was ligated into a *pGEM-T* vector (Figure 5.1.2a *(iv)* and *(v)*) and transfected into JM109 cells (plasmids, *pS2b* and *pS2c*). DNA was extracted by mini-prep after selection through blue/white screening. The orientations of the inserts were determined by PCR using primer pairs from the following: $long_L$, $long_R$ and 5'-ATTTAGGTGACACTATAG-3' (*sp6*). A small section was cut out of the plasmids in both orientations by digestion with *Nhe*I plus *Sac*II (Figure 5.1.2b *(i)*) and *Avr*II plus *Sac*II (Figure 5.1.2b *(ii)*). The two results were gel extracted and each was ligated separately with the *bfp* extract. The ligation products were transfected and this selected for the target constructs containing a "longmer" and *bfp* gene (plasmids, *pS2d* and *pS2e*): The *bfp* extract and each "longmer" extract (Figure 5.1.2b *(i)* to *(iii)*) cannot circularise themselves as the two cohesive ends of each individual extract are incompatible. A single origin of replication is required to allow a plasmid to be replicated efficiently and an *ampr* gene provides resistance to host cells when incubated with ampicillin. A plasmid formed by the ligation of a *bfp* extract and one of the "longmer" extracts provides both ampicillin resistance and a single origin of replication (Section 5.3.2).

5.1.3 Stage 3

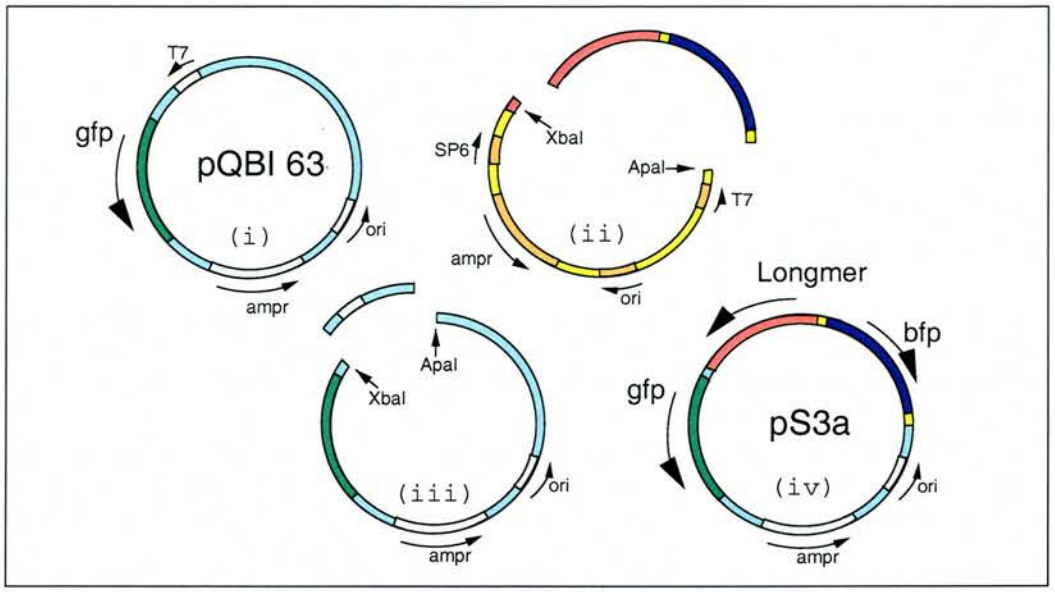


Figure 5.1.3a: Plasmids constructed during Stage 3*a*. (i) GFP expression plasmid, *pQBI 63* (QBiogene). *gfp* gene shown in dark green. (ii) Plasmid *p2Se* after digestion with *Xba*I and *Apa*I. (iii) Plasmid *pQBI 63* after digestion with *Xba*I and *Apa*I. (iv) A longmer oligonucleotide upstream of a *gfp* gene with a *bfp* gene oriented in the opposite direction inside the plasmid backbone of *pQBI 63*.

The plasmid, *pQBI-63*, was transfected into competent JM109-DE3 cells which express T7 RNA polymerase when incubated with IPTG. The plasmid DNA produced from a mini-prep had a small section removed upstream of the *gfp* gene using *Xba*I and *Apa*I, followed by its gel extraction. *pS2e* was also digested with *Xba*I and *Apa*I to allow its insert to be gel extracted and then ligated with the *gfp* extract.[†] The resulting DNA was transfected into JM109-DE3 cells and this selected (Section 5.3.2) for the target constructs containing a "longmer" and both AFP genes (plasmid, *pS3a*).

[†]Initially, it was intended that the *bfp* gene would be digested at its *Nhe*I restriction site and inserted at the longmer's *Avr*II restriction site. The resulting construct carried in the *pGEM-T* vector could not be cloned into JM109. Furthermore, the results of combining *pS2d* with *gfp* could also not be cloned. However, it was found in the second stage that a chimeric *bfp*, in *pS2e*, caused by digestion at *bfp*'s *Xba*I restriction site produced comparable fluorescence to *pQBI-T7-BFP*. Therefore, *pS2e* was used instead, and the *bfp* gene was digested further upstream using *Xba*I.

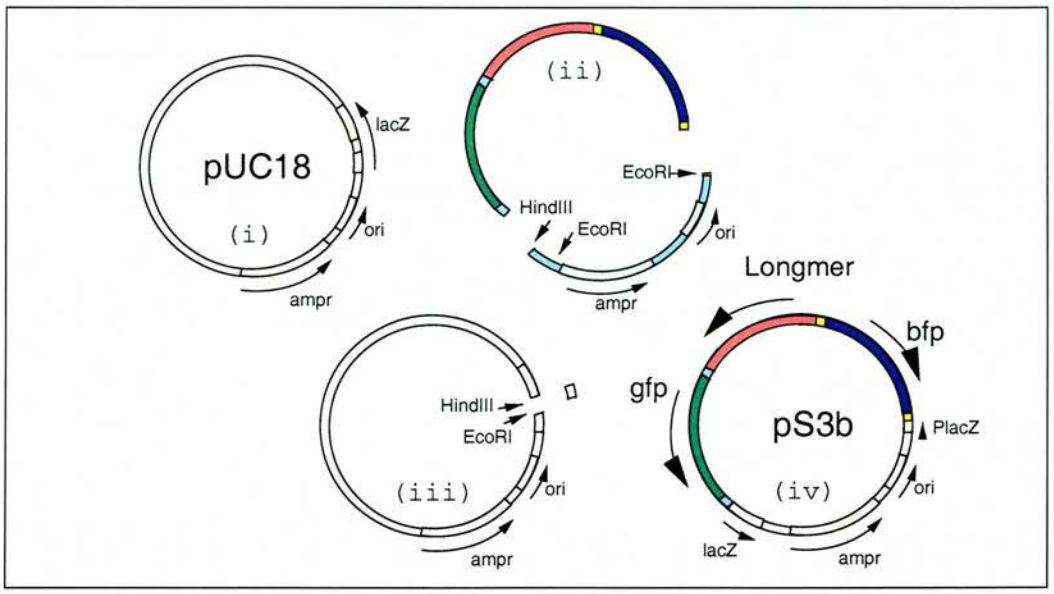


Figure 5.1.3a: Plasmids constructed during Stage 3*b*. (i) The *pUC18* cloning vector. (ii) Plasmid *pS3a* after digestion with *Hin*dIII and *Eco*RI. (iii) Plasmid *pUC18* after digestion with *Hin*dIII and *Eco*RI in its cloning region. (iv) A *gfp* gene upstream of the longmer oligonucleotide with a *bfp* gene in the opposite direction on a *pUC18* backbone.

To remove an additional *Bpu*10I restriction site in *pS3a* from *pQBI-63*, the plasmid, *pUC18*, was transfected into JM109 cells, and the DNA plasmid produced from a mini-prep was digested with *Hin*dIII and *Eco*RI in its cloning region, followed by a gel extraction of the linearised plasmid. DNA from *pS3a* was also digested with *Hin*dIII and *Eco*RI, and the section containing the AFP genes was gel extracted. The two extractions were ligated together and the resulting DNA was transfected into JM109-DE3 cells, selecting for the target construct (plasmid *pS3b*). Both extracts, with cohesive ends produced by digestion with *Hin*dIII and *Eco*RI cannot recircularise themselves, and only one extract contains an *ampr* gene and origin of replication (Section 5.3.2).

5.2 Rotation

5.2.1 Stage 1

The first experiment aimed to rotate a promoter, which was flanked by two restriction sites (Section 5.1.1), using a restriction endonuclease and a ligase. The purpose of this experiment was to investigate the plausibility of a DNA switch that uses a mech-

anism driven by a restriction endonuclease. Given the results of the earlier simulation experiments of a DNA switch (Section 4.2.3), it was predicted that, after exposure to *Bpu*10I and a ligase, the direction of the promoter in the switch would be inverted, and that the promoter would be flanked by two new sequences which were not recognised by *Bpu*10I. The rotation operation in this stage was performed by a series of separate digestions and ligations of plasmid, *pS1*, to create plasmid, *pS1-R*.

a.	Digestion of *pS1* with *Bpu*10I in *Bpu*10I unique buffer.
b.	Ligation with T7 DNA Ligase in T7 ligation buffer with 10% PEG 4000, cleanup.
c.	Amplification of insert using primers, $long_R$ and $long_L$, to increase DNA quantity.
d.	Gel extraction of DNA around 200bp.
e.	Digestion with *Bpu*10I in *Bpu*10I unique buffer to select for *pS1-R*, cleanup.
f.	Gel extraction of DNA around 200bp.

Data was collected by transfecting the rotated product, *pS1-R*, and then sending the resulting DNA for sequencing (Section 5.3.1) after preliminary digestion tests.

g.	Transfect extraction into JM109 (*pS1-R*), grow, mini-prep.
h.	Linearise DNA with *Sal*I or *Sac*II (insert orientation dependant), cleanup.
i.	Gel verification by digestion with *Bpu*10I in *Bpu*10I unique buffer.
j.	Send for sequencing using primers, $long_R$ and $long_L$.

5.2.2 Stage 2

The purpose of the second experiment was to investigate the behaviour of DNA switches that report their state *in vivo*. Two plasmids, *pS2d* and *pS2e*, were constructed, each containing a rotatable promoter that was flanked by two *Bpu*10I restriction sites (Section 5.1.2). Plasmid *pS2d* contained a *bfp* gene downstream of the promoter and it was expected that cells with this plasmid would express more BFP before the rotation of the promoter than afterwards. Conversely, plasmid *pS2e* contained a *bfp* gene upstream of the promoter, and it was predicted that this plasmid would cause the expression of less BFP before the rotation than afterwards. The rotation operations for the second stage used DNA from plasmids, *pS2d* and *pS2e* to create plasmids *pS2d-R* and *pS2e-R*, respectively. A single buffer was used at 22°C after exploratory experiments found that the *Bpu*10I enzyme was active at this temperature in the presence of PEG 4000 (Section B.2.3).

Plasmid DNA ($0.4g\ell^{-1}$)	$3\mu\ell$	
PEG 4000 (50%)	$4\mu\ell$	
T4 ligation buffer	$3\mu\ell$	a. Incubate at 22°C for 4 hours.
*Bpu*10I enzyme	$2\mu\ell$	
T4 DNA Ligase	$1\mu\ell$	
H_2O	$10\mu\ell$	

Data was collected by transfection of the products into T7 RNA polymerase expressing cells that were activated by IPTG-induction and then viewed under a microscope (Section 5.3.3).

b.	Deactivation by 10 minutes at 65°C, followed by transfection into JM109-DE3.
c.	Spin down resulting culture at 13rpm for 3 minutes.
d.	Resuspend in $5m\ell$ of LB solution with ampicillin and grow overnight at 37°C.
e.	Subculture $1m\ell$ of overnight solution in $5m\ell$ and incubate at 37°C for 4 hours.
f.	Add $12\frac{1}{2}\mu\ell$ 0.1M IPTG and incubate at 30°C for 4 hours.

5.2.3 Stage 3

A plasmid was constructed for the third stage that contained a promoter flanked by two *Bpu*10I restriction sites, with a *gfp* gene downstream of the promoter and a *bfp* gene upstream of the promoter (Section 5.1.3). Like the second stage, the purpose was to investigate the behaviour of a DNA switch that reports its state *in vivo*. However, rather than measuring the expression of a single gene, by making a comparison of the expression of two different genes, it was predicted that this would provide a clearer indication of the state of a switch. It was expected that DNA switches before a rotation operation would cause the expression of GFP, and, that after rotation, BFP would be synthesised instead. It was also predicted that the leaky expression of the two reporter genes would be considerably lower than that found in the second stage (Section 5.3.3) due to the considerations made during the construction of *pS3b* (Section 5.1.3). This stage made use of a common buffer for each *in vitro* reaction on plasmid, *pS3b*, that was incubated at room temperature (22°C) to create plasmid, *pS3b-R*.

PEG 4000 (50%)	$4\mu\ell$
T4 ligation buffer	$2\mu\ell$
H_2O	$2\mu\ell$
common buffer	$8\mu\ell$

An extra cleanup stage and late addition of *Bpu*10I enzyme were performed in an attempt to increase the yield of *pS3b-R*.

a.	Incubate 10$\mu\ell$ *pS3b*, 8$\mu\ell$ buffer and 1$\mu\ell$ *Bpu*10I for 1 hour and cleanup.
b.	Incubate 10$\mu\ell$ of result with 8$\mu\ell$ buffer and 1$\mu\ell$ T4 DNA ligase for 6 hours.
c.	Continue to incubate for a further hour after adding 1$\mu\ell$ *Bpu*10I.

Reporting

The extent of rotation (Section 5.3.4) was determined by transfecting the resulting products into T7 RNA polymerase expressing cells and smearing sample mixtures onto ampicillin plates. It was expected that *pS3b* colonies would fluoresce green because of the *gfp* gene that lies downstream of the promoter. It was also expected that some *pS3b-R* colonies would fluoresce blue (indicating a rotated promoter) and some colonies would fluoresce green (indicating an unrotated promoter).

d.	Transfection into JM109-DE3 (no deactivation).
e.	Spread 200mℓ of resulting culture to grow overnight at 37°C on ampicillin plates.
f.	Leave plates to incubate at room temperature for 6 hours to increase fluorescence.

Stability

The stability of *pS3b* and *pS3b-R* colonies in retaining their respective green and blue characteristics over generations of bacteria was determined (Section 5.3.5) by picking and growing up individual colonies from plates from the construction stage of *pS3b* and sub-stage (f) for *pS3b-R*. Then, a sample of the colonies was smeared on ampicillin plates. As the state of each switch was stored in DNA, it was expected that the state would be retained over generations of bacteria.

g.	Pick colonies from plates and inoculate 5mℓ of LB with ampicillin.
h.	Grow overnight at 37°C and make glycerol stocks.
i.	Inoculate 5mℓ LB solution + amp with glycerol stock scraping, grow overnight.
j.	Spread 50$\mu\ell$ of the mixtures on ampicillin plates and grow overnight at 37°C.
k.	Leave plates to incubate at room temperature for 6 hours to increase fluorescence.

Distinction

The distinction between the signals reported by *pS3b* and *pS3b-R* was determined (Section 5.3.6) by taking fluorimeter readings with picked colonies that had been grown up and left at room temperature. As one of the aims of the construction of the plasmid *pS3b* was to minimise leaky transcription (Section 5.1.3), it was expected that the fluorescent colours of the colonies with *unset* and *set* switches could be clearly distinguished: it was predicted that there would be little overlap in the emission spectra of colonies from before and after the rotation operation.

l.	About 5mℓ of cultures from (i) were incubated at room temperature for 24 hours.
m.	The results were scanned for emissions with an excitation peak of 387nm.

Irreversibility

Finally, the irreversibility of the rotation operation was determined (Section 5.3.7) by linearising plasmids, *pS3b* and *pS3b-R* and then digesting both with *Bpu*10I. As *pS3b-R* was predicted to contain no *Bpu*10I restriction sites (Section 4.2.3) it was expected that its DNA could not be digested with this restriction endonuclease. However, *pS3b* contains two *Bpu*10I restriction sites and so can be digested with *Bpu*10I.

n.	Plasmid DNA from *pS3b* and *pS3b-R* was extracted by mini-prep
o.	The resulting DNA was linearised with *Xba*I and then digested with *Bpu*10I

5.3 Results

5.3.1 Rotation

A plasmid, *pS1*, containing a bacteriophage T7 promoter flanked by *Bpu*10I restriction sites (Section 5.1.1) was subjected to a rotation operation, comprising a series of digestion and ligation cycles (Section 5.2.1), attempting to invert the direction of this T7 promoter. DNA, around the promoter and flanking sites, from before (*pS1*) and after (*pS1-R*) the rotation operation, was sequenced to determine the outcome.

```
pS1:     CTAGCCATA TGTATATCTC CTTCTGCTTA GGATGCGTCC GGCGTAGAGG ATCGAGATCT CGATCCCGCG
         ||||||||| |||||||||| |||||
pS1-R:   CTAGCCATA TGTATATCTC CTTCTGCTTA GCCGCTCAxA ATTCCCCTAT AGTGAGTCGT ATTAATTTCG

pS1:     AAATTAATAC GACTCACTAT AGGGGAATTx TGAGCGGCTA AGGATTCCCT CCTAG
                                                   ||||||| |||||
pS1-R:   CGGGATCGAG ATCTCGATCC TCTACGCCGG ACGCATCCTA AGGATTCCCT CCTAG
```

Figure 5.3.1: Comparison of sequences before rotation, *pS1*, and after rotation, *pS1-R* (top and bottom respectively). *Bpu*10I sites are underlined and those bases which did not take part in the rotation operation are connected with a '|'. A single base pair deletion occurred during the construction of *pS1* which is marked with an 'x'. Sequencing was performed by DNA Sequencing Services, Department of Biological Sciences, Durham University.

After the rotation operation had been performed, the 80bp mid-section of the "long-mer", in between the two *Bpu*10I sites, 5'-GCTTAGG-3' and 5'-GCTAAGG-3', was inverted (Figure 5.3.1). The *Bpu*10I site sequences were replaced with 5'-GCTTAGC-3' and 5'-CCTTAGG-3', respectively, neither of which matches the *Bpu*10I recognition template of 5'-CCTNAGC-3'. Sequencing also showed that a single base pair deletion had occurred at the same time during the construction process, but was in a region that does not affect the transcriptional or rotational machinery.

The result of this experiment demonstrates a section of DNA being rotated through the combined actions of a restriction endonuclease and ligase.

5.3.2 Selection

During the construction of the stage two plasmids (*pS2d* and *pS2e*), it was found that by ligating a linearised vector holding an insert with dsDNA excised from a vector with the same backbone, *both* having cohesive ends of *Sac*II (5'-GC-3') or *Avr*II, and *Xba*I or *Nhe*I (3'-CTAG-5'), followed by transfection into competent JM109 cells, this selected for plasmids containing the insert (Figure 5.3.2). This was found to be a convenient way to make a series of inserts, held in the same vector, ligate together efficiently.

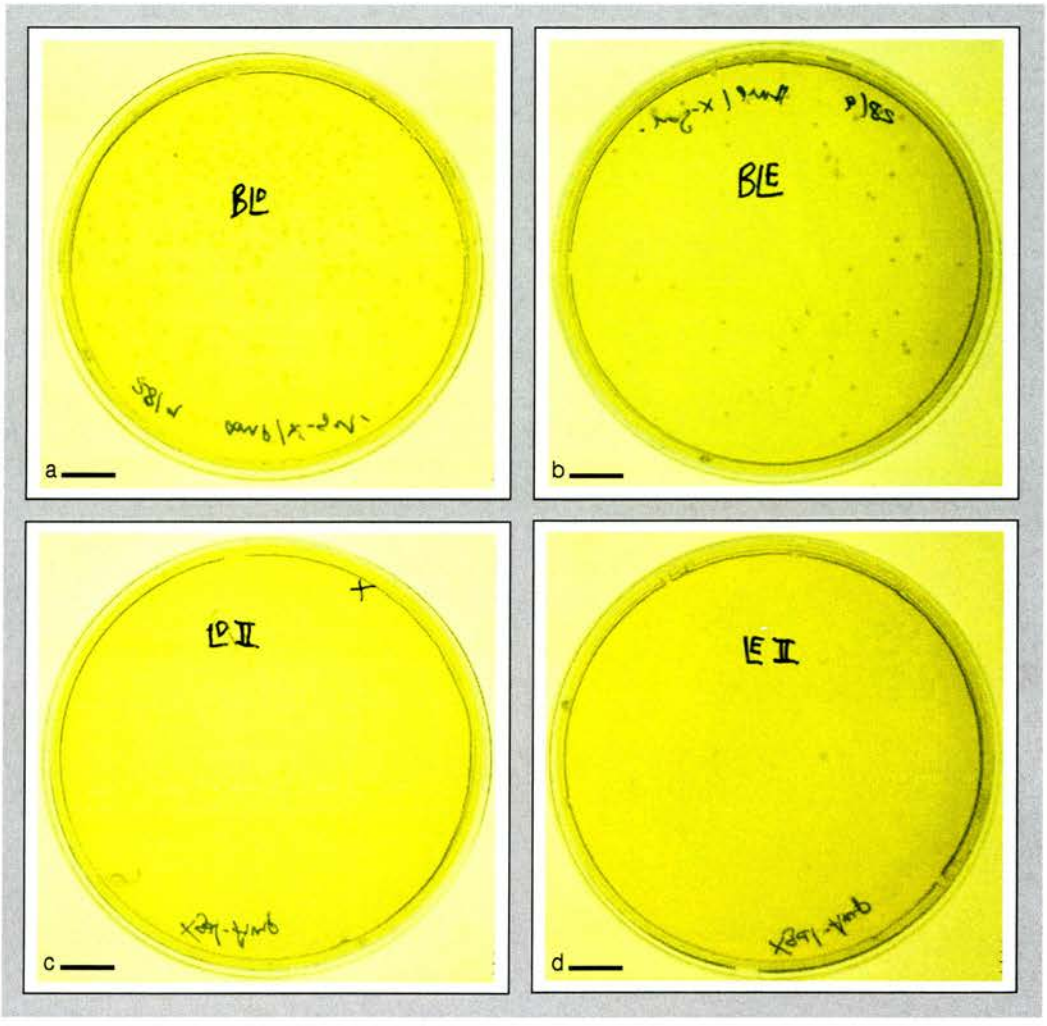


Figure 5.3.2: Transfection results of ligated plasmid sections. Plasmid *pS2a* extract and broken plasmid *pS2b* produced more than 50 colonies (a) as did *pS2a* and *pS2c* (b). However plasmid *pS2b* ligated alone produced 2 colonies (c) and plasmid *pS2c* ligated with itself produced no colonies (d). Scale bar: 10mm.

The linearised plasmids contain an origin of replication and gene conferring ampicillin resistance. All four picked colonies for *pSd* and *pSe* showed inserts of expected size when digested with *Eco*RI. Furthermore, ligating together broken plasmids produced a very low colony number. The size of the insert of the other two control colonies, when digested with *Eco*RI, could not be determined. This is possibly because the plasmid DNA from these colonies between *Eco*RI restriction sites was too small to be detected using gel electrophoresis or because DNA had been mutated, after transfection, and this process had removed *Eco*RI restriction sites.

It is expected that a broken plasmid ligated to a copy of itself would be selected against owing to the two resulting origins of replication. And, given the importance of having ampicillin resistance when growing in media with ampicillin, the optimum plasmid is one consisting of a single broken plasmid, rejoined with the extracted insert. This method of selection was also used successfully for two other construction stages, (Section 5.1.3), with different plasmids.

5.3.3 Noise

Two plasmids were constructed for this experiment (Section 5.1.2): one with a T7 promoter pointing towards a *bfp* gene (plasmid *pS2d*), and one with a T7 promoter pointing away from a *bfp* gene (plasmid *pS2e*). Both promoters were flanked by *Bpu*10I recognition sequences to allow their inversion. It was expected that T7 RNA polymerase-expressing JM109 DE3 cells carrying *pS2d* would fluoresce blue strongly before rotation (Section 5.2.2), and very little after rotation, and vice-versa in the case of *pS2e*. However, owing to the discovery of leaky transcription and wide variations in the amount of BFP produced for the same plasmid (Figure 5.3.3), this prompted extensions to this experiment that were used in stage three (Section 5.1.3).

plasmid	x_0	x_1	x_2	x_3	μ	σ_{n-1}
pS2d	0.74	0.50	0.83	0.61	0.67	0.15
pS2d after rotation	0.03	0.03	0.03	0.10	0.05	0.04
pS2e	0.48	0.35	0.64	0.30	0.44	0.15
pS2e after rotation	0.18	0.21	0.42	0.49	0.33	0.15
QBI-T7-BFP	0.30	0.28	0.42	0.32	0.33	0.06

Table 5.3.3: The proportion of bacteria that fluoresced with at least half of their total visible area when subjected to UV light, viewed with a Leica DC350F camera connected to a Leitz Orthoplan epifluorescence microscope. The total area of a cell was determined by counting the number of pixels inside, and including, the cell membrane when viewed with white light. Four sample readings were taken per IPTG-incubated colony and white light images were digitally overlaid with UV light images to determine proportion values.

There was a significant decrease in fluorescence when *pS2d* was subjected to the rotation operation as expected with a standard deviation (SDEV) of more than 3. Furthermore, *pS2d* after rotation showed a significantly lower fluorescence than plasmid *QBI-T7-BFP* (more than 2 SDEV). No significant difference was found between readings of the three plasmids: *QBI-T7-BFP* and *pS2e* before and after rotation (less than 1 SDEV). Unexpectedly, this result suggests that plasmid *pS2d* produces *more* fluorescence than plasmid *QBI-T7-BFP* (more than 1 SDEV).

When it was discovered that plasmid *pS2a* in JM109 DE3 cells, containing only a *bfp* gene in a *pGEM-T* plasmid, fluoresced under UV light despite being oriented away from the vector's T7 promoter, it was concluded that some aspect of the *pS2d* and *pS2e* constructs was causing leaky and additional transcription of the *bfp* gene. T7 RNA polymerase is likely to have begun transcription from either the *pGEM-T* SP6 site or *E. coli* promoter from the disrupted *lac*Z gene. The third construction stage (Section 5.1.3) sought to correct this problem: by moving to a vector without potentially interfering promoters and placing *bfp* and *gfp* genes either side of the "longmer". Having two encodings to report each state allows relative *before* and *after* readings to be taken, and also has the side effect of reducing the chance of leaky transcription from the vector.

5.3.4 Reporting

The plasmid, *pS3b*, was constructed so that a T7 promoter lies upstream of a *gfp* gene (Section 5.1.3). The T7 promoter is flanked by *Bpu*10I restriction sites allowing the rotation of the sequence in between. Once rotated, the T7 promoter points towards a *bfp* gene, rather than the *gfp* gene. This plasmid was subjected to a rotation operation (Section 5.2.3), and *in vivo* expression of the AFPs, before and after rotation, was measured.

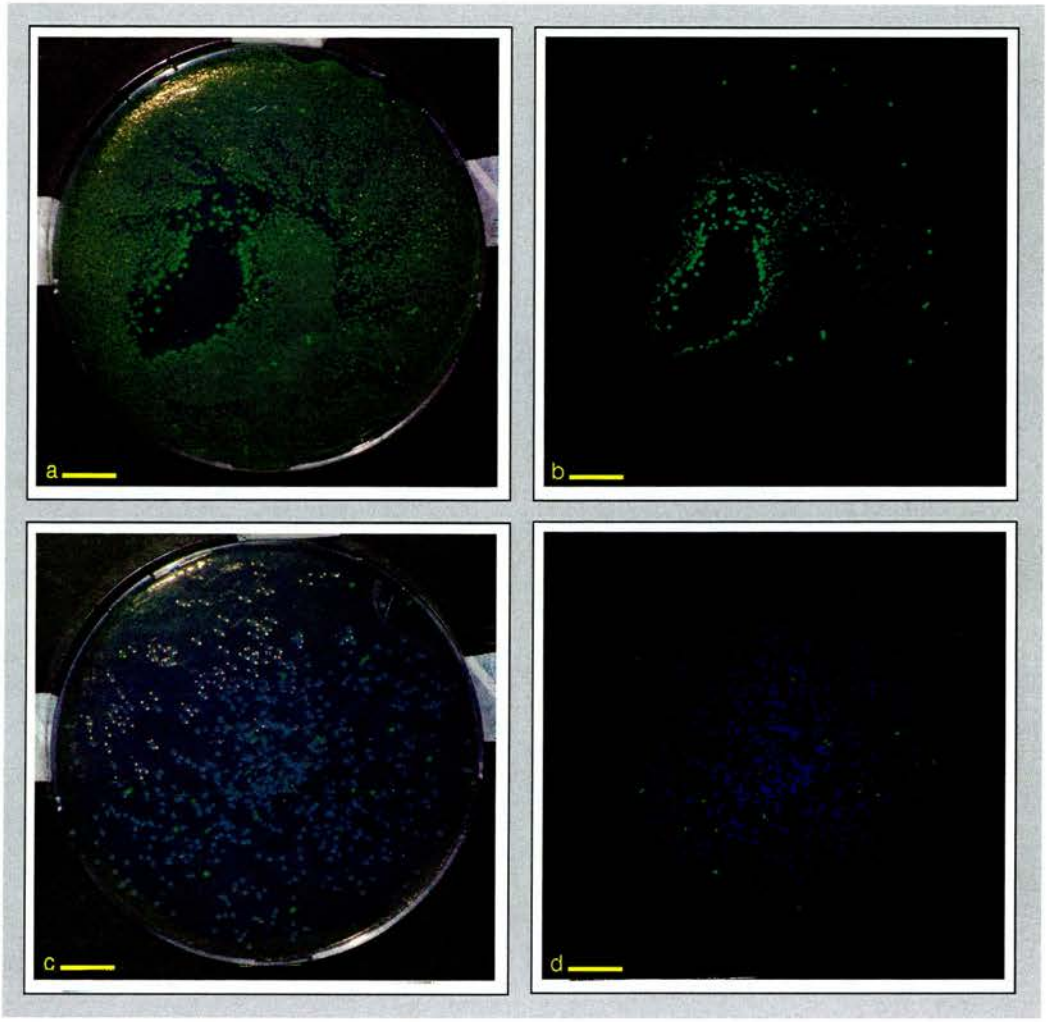


Figure 5.3.4: The results of transfecting *pQBI63* section with *pS2e* under white light (a) and UV light (b) when constructing plasmid *pS3b*. After the rotation operation, a small proportion of unconverted green colonies can be seen under white light (c) and the majority glow blue under UV light (d). Plates were smeared with 200$\mu\ell$ of transfection mixture. Photographs were taken with a Fuji S602 digital camera, UV light was supplied by an IBI transilluminator and white light by the camera's flash. Scale bar: 10mm.

The majority of colonies resulting from the ligation of the excised DNA from *pS2e* (holding a "longmer" pointing away from a *bfp gene*) and *pQBI63* (Section 5.1.3) glowed green under white light (Figure 5.3.4a) and UV light (Figure 5.3.4b). A small number of colonies can be seen to be glowing cyan (green and blue) under UV light. It is possible that a string of concatenated *pS2e* extracts were ligated to a broken *pQBI63* plasmid, thus allowing a T7 promoter to point towards a *bfp* gene.

After the rotation operation on a single green colony picked from the plate (5.3.4a,b), the resulting sample of transfection mixture showed the majority of cells glowing blue under UV light (Figure 5.3.4d) and a small proportion of unrotated cells glowing green under white (Figure 5.3.4c) and UV light. Similar results were obtained when applying the rotation operation to three other colonies from the source plate. Colonies from "before" and "after" were also viewed with a microscope (Figure 5.3.4ii) which showed clearly distinguishable states.

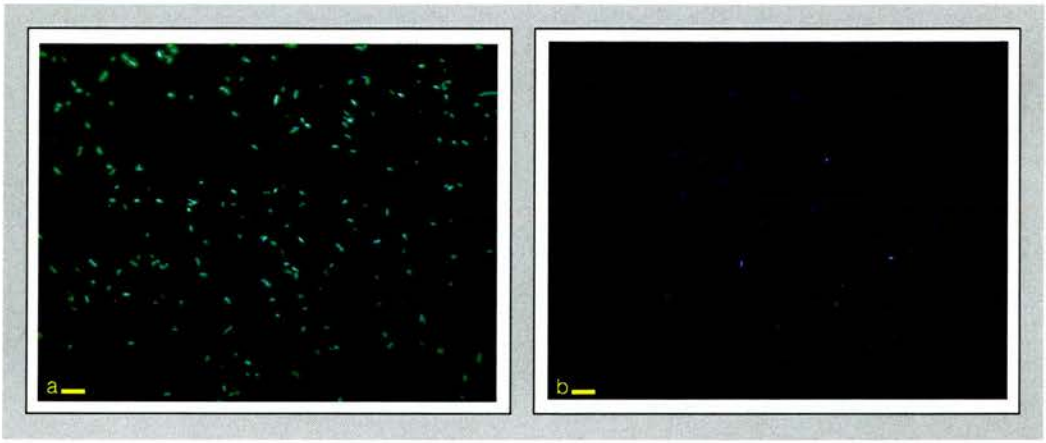


Figure 5.3.4ii: An illustration of bacteria from before and after rotation (a and b respectively) as viewed with a Leica DC350F camera connected to a Leitz Orthoplan epifluorescence microscope. Monochrome images for white light emission and UV light emission were digitally combined with green and blue filters respectively. Scale bar: 10μm.

5.3.5 Stability

To assess the stability of the state of the green-blue DNA switch, colonies, of both before and after the rotation operation had been performed, were grown through two overnight cycles (Section 5.2.3, *Stability*) and their green and blue fluorescence was measured (Figure 5.3.5).

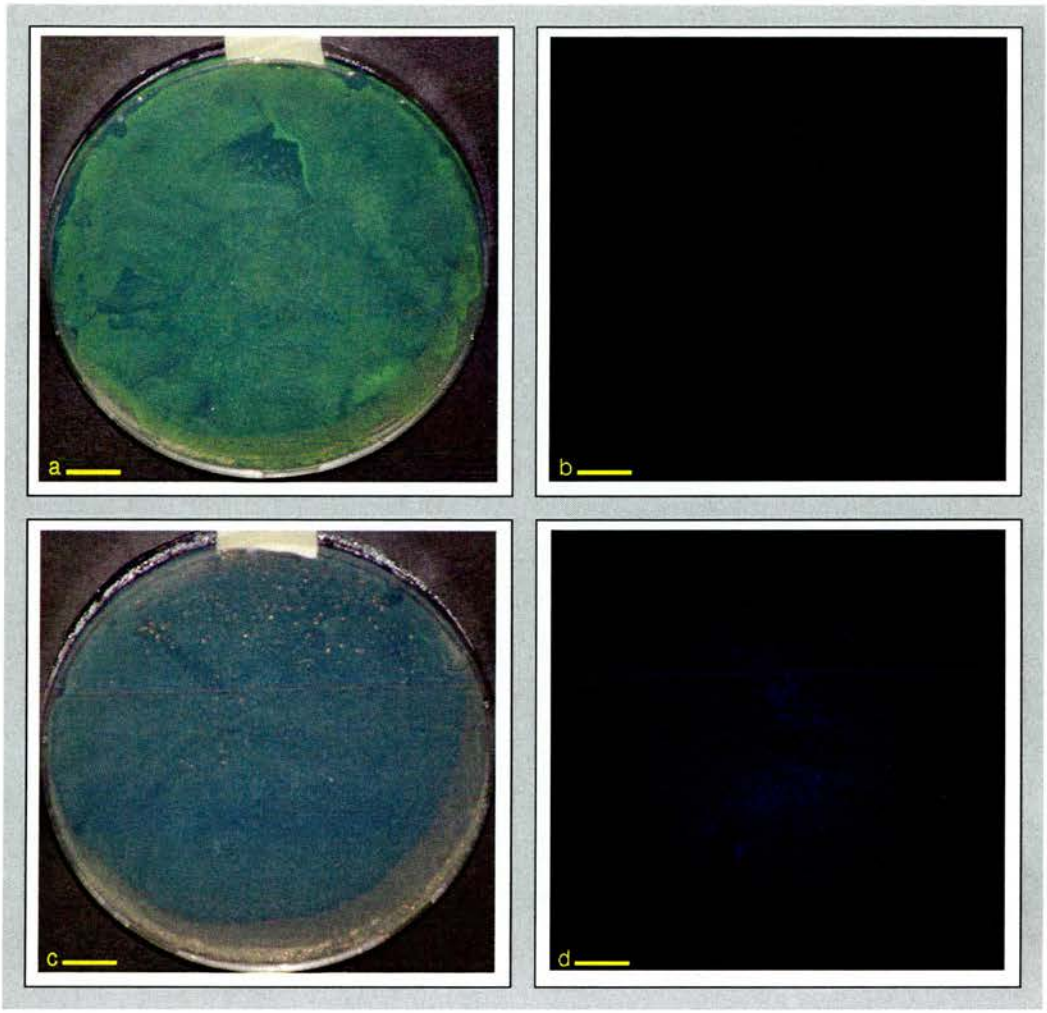


Figure 5.3.5: The results of the repeated growing up of single colonies picked from plates from before and after rotation, smearing 50$\mu\ell$ of solution on each plate. a: *pS3b* under white light. b: *pS3b* under UV light. c: *pS3b-R* under white light. d: *pS3b-R* under UV light. Scale bar 10mm.

After repeated growing up of colonies from before and after rotation, the green and blue fluorescences, respectively, were retained over generations of bacteria. This test was carried out four times with similar results.

5.3.6 Distinction

Colonies of JM109 DE3 cells carrying *pS3b* and *pS3b-R* plasmids were subjected to UV excitation and the resulting emissions were measured (Section 5.2.3, *Distinction*).

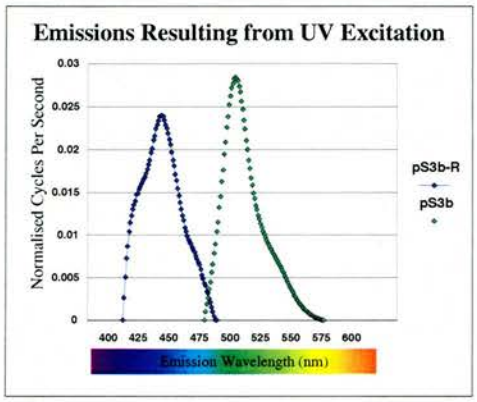


Figure 5.3.6: Emissions of colonies containing plasmids, *pS3b* and *pS3b-R*. An excitation wavelength of 387nm was used and emission readings between 400nm and 625nm were recorded at 1nm and then subtracted from using background readings from a control sample of JM109 cells with similar density. Readings were taken with a SPEX FluoroMax 5030 (Instruments SA) and were normalised by the sum of positive readings for each culture to present two relative signatures for before and after rotation, as shown.

The emission signatures of colonies from before (*pS3b*) and after (*pS3b-R*) rotation showed distinct green and blue readings respectively when excited with UV light (Figure 5.3.6).

5.3.7 Irreversibility

To show that a switch's state, once set, is irreversibly stored in DNA, digestions were performed on plasmids *pS3b* and *pS3b-R* (Section 5.2.3, *"Irreversibility"*).

Figure 5.3.7: Digestion results of "before", *pS3b* and "after", *pS3b-R* plasmids.a: *pS3b* linearised with *Xba*I. b: linearised *pS3b* digested with *Bpu*10I. c: *pS3b-R* linearised with *Xba*I. d: linearised *pS3b-R* digested with *Bpu*10I.

The plasmid DNA of *pS3b* (before rotation) could be digested with *Bpu*10I however, the DNA of *pS3b-R* (after rotation) could not be digested with *Bpu*10I (Figure 5.3.7). Therefore, *pS3b* is susceptible to the rotation operation, whereas *pS3b-R* is not.

Summary

This chapter has described how synthetic DNA switches were constructed, how a rotation operation was performed on them and how measurements were taken. The next chapter presents and discusses the resulting behaviour of the DNA switches. The results of testing synthetic DNA switches in wetware were then presented. The next chapter discusses the consequences of these experimental results, and the simulation results in the context of related work, and future avenues of exploration are proposed.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis has investigated the concept and plausibility of running DNA programs inside bacteria which, when triggered by external stimuli, can be rearranged in a directed manner. In order to explore and demonstrate the potential of reprogrammable DNA programs, software was created to search, simulate and evaluate designs for DNA programs which are able to reassemble themselves. The unconstrained space of DNA programs is potentially vast and to overcome this, an approach was adopted towards the notational representation of DNA programs and a search strategy employed. As a proof of concept, a design found by the software, a synthetic DNA switch, was implemented in wetware to demonstrate its biological plausibility.

When modelling DNA rearrangement based on enzymatic recognition sites, a minimal level of abstraction is the nucleotide base sequences which compose those sites because one of the side effects of a DNA rearrangement may be the creation of a new recognition site with a new sequence, which may affect subsequent potential rearrangements. However, it was unreasonable to model DNA programs entirely at this level as it is often not possible to infer the functionality of an arbitrary length of DNA based on its sequence. Consequently, a hybrid representation was adopted which combines nucleotide base sequences with the behavioural functionalities of genetic components.

A second problem encountered is that there is a limited number of genetic components which have been well-characterised and which are readily available for incorporation into DNA programs. Consequently, the search strategy draws from a database of *prede-*

fined components, rather than allowing the search engine to generate components with properties that contribute optimally to search criteria [François and Hakim, 2004]. One advantage of this approach is that the generated DNA programs are partly expressed in symbols representing known DNA components. Consequently, DNA programs generated could be readily compared and verified against existing natural and synthetic DNA programs. A second and more basic advantage was the considerable reduction in the size of a search space which does not include component properties. A disadvantage to the use of the component database was that despite the availability of databases of qualitative genetic site properties [Salgado *et al.*, 2004], [Keseler *et al.*, 2005], the quantitative properties had to be derived from the literature rather than linked to a standardised database. To partly counter this, the component database was made up of components with a range of numerical properties and all such properties were used as probabilistic reaction thresholds. Furthermore, the DNA programs were also simulated in a noisy environment of typically 1 to a few tens of particles per signal, thus reducing the possibility of generating a DNA program that's operation was dependent on particular parameter values.

With the experimental criteria presented in this thesis, the search software is able to rank a search space of DNA programs composed of up to 15 genetic sites from a database of over 40 sites (with some sequences allowing ambiguity), within one day on a modern desktop computer (3GHz processor). A simple brute force approach towards traversing a DNA program search space is impractical; therefore, a collection of DNA program design rules was created to allow a search space to be *structured* by behavioural criteria provided by an experimenter. Pre-simulation pruning heuristics were also employed, where possible. An advantage of this technique, which was sufficiently efficient to allow complete search space traversals, is that the search algorithm will not miss functionalities provided by the combination of a set of components which in isolation provide no benefit, in contrast to a heuristic search. However, it could reasonably be argued that the design rules themselves could hide potentially optimal points in a search space. But, the design rules can and have been expressed explicitly and their *limitations* can be directly criticised and improved. It is arguably more difficult to explain and assess the limitations of an approach which uses a heuristic *process* (e.g. evolutionary search algorithms [Foster, 2001]) in order to traverse a search space.

This is related to a universal problem when using computer software to generate solu-

tions, *viz.* transparency. It is important that the software should be able to justify its choices. The approach taken here allows the *input* rules to be specified in behavioural terms, and for the *output* choices to be justified with direct reference to the input rules. Furthermore, for some of the simulated rearrangements of DNA programs found, it was not immediately obvious how the *input* DNA program of a simulation caused the *output* RNA concentration and DNA composition results. This problem was tackled by producing a booklet for each simulation run which graphically represented the regulatory and rearrangement effects that a DNA program is subjected to over time, which can be referenced against its output results. These representations were based on the symbolic notation of sites using two levels of abstraction, described earlier. As most DNA sites are represented by symbols, a reader familiar with their associated properties can quickly identify the content of a DNA program in a manner similar to looking at a concise biological gene line.

The software created served two purposes towards the aims of this thesis. On the one hand, it has produced designs of potential DNA reprogrammability operations for implementation in wetware. On the other hand, it has been used to demonstrate how a cell can be triggered, through gene regulation, to rearrange a DNA program in a directed manner, and to illustrate the potential scalability of rearrangement operations when performed cooperatively.

In contrast to a synthetic *RNA* switch [Gardener *et al.*, 2000], a two-colour *DNA* switch was implemented in wetware based on one of the designs produced by the software. The DNA switch was able to demonstrate the plausibility of a rotational reprogramming operation on a DNA program using *in vitro* manipulations, and its stability and clarity of reported signal was confirmed when operating *in vivo*. The enzymes used to drive the rotation operation, *Bpu*10I and T4 DNA ligase, had previously been cloned and sequenced - Stankevicius *et al.* demonstrated regulated *Bpu*10I expression in *E. coli* [Stankevicius *et al.*, 1998], and the ligase is natively expressed by the virus, T4, when it infects bacteria, such as *E. coli*. Experiments leading to a successful wetware demonstration highlighted one of the particular difficulties of implementing synthetic devices; owing, in part, to high levels of intrinsic transcriptional noise in *E. coli* [Elowitz *et al.*, 2002] found in early experiments, the DNA switch had to be redesigned to report its state using two reporter proteins to allow more robust, relative measurements of its state.

6.2 Summary of Contributions

1. **Notation:** A symbolic notation has been presented which can be used to describe DNA programs as concise lines of DNA units, a convention familiar to biologists, and their associated interactions with proteins. The notation supports two levels of abstraction: regulatory units and nucleotide base sequences in order to allow efficient symbolic manipulation, while also supporting simulated DNA program rearrangements.

2. **Simulator:** The "DNA Program Simulator" and its platform, BioSim which can execute a DNA program written in a symbolic notation have been described. The simulator can dynamically map between a DNA program and its implicit, emergent genetic network such that stochastic, simulated gene regulatory events can trigger the reprogramming of a DNA program. Such a reprogramming operation results in a genetic network being reassembled *during* its own execution. To aid the understanding and verification of sometimes complex regulatory and rearrangement processes, the simulator produces a *story booklet* composed of pictorial snapshots of DNA programs which are taken automatically around the time of DNA arrangement events and changes in regulatory influences. These snapshots are also placed alongside plots of RNA concentrations over time.

3. **Search:** The "DNA Program Generator" combines rational design heuristics with a search algorithm to create DNA programs, composed of sites drawn from a database, which fulfil a set of behavioural criteria. Efforts have been made to allow criteria to be expressed in a rule-based manner which is accessible to those unfamiliar with programming. DNA programs, expressed in notation, are simulated to obtain tables of DNA composition and RNA concentrations over time, which enables the automated evaluation and ranking of the generated DNA programs. Furthermore, the generator provides *justifications* for its choices of DNA programs to aid transparency and the verification of its methods.

4. **DNA Switch:** One of the DNA programs found by the generator – a two colour DNA switch – was implemented in wetware. The rearrangement of the synthetic DNA switch was driven, in part, through the use of a restriction endonuclease *in vitro*, while measurements were observed *in vivo*. In addition to demonstrating the *plausibility* of a DNA reprogramming operation, the following were also demonstrated: the stability of the DNA switch across generations of bacte-

ria, that the DNA switch's state can be clearly distinguished, and that the DNA switch's state can be set irreversibly.

5. **Scalability:** The computational scalability of DNA reprogramming operations that are mediated by restriction endonucleases was investigated. It was demonstrated, through simulations, how a DNA reprogramming operation can be controlled using gene regulation and triggered by external signals. It was also shown how different restriction endonucleases can be used, in combination, to cause directed and cooperative rearrangements of DNA by exploiting the modularity of comparable cohesive ends.

6.3 Areas of Future Work

6.3.1 *In Vivo* DNA Storage and Reprogrammable Library

It will be instructive to create and assess the characteristics of an entirely *in vivo*, reprogrammable memory system in wetware based on one or more DNA switches which are controlled by external signals, such as specific wavelengths of light [Shimizu-Sato *et al.*, 2002]. The mechanisms of such a DNA memory system have been demonstrated in isolation and researchers in the field of synthetic biology are advancing our ability to integrate synthetic components together [Yokobayashi *et al.*, 2002]. The memory system could be used to demonstrate a number of possibilities: from the recording of environmental events to the directed manipulation of a library of components, causing their rearrangement to form new DNA programs. Given the current, and sometimes difficult to predict, practical limitations on the size and composition of a DNA program that can be added to a cell, it will also be interesting to explore the possibility of addressing this problem by expanding the storage of information across different cells, such as by utilising a synthetic intercell communication mechanism [Weiss *et al.*, 2003].

6.3.2 Simulation Features

The simulation and search software could be enhanced in several ways. Firstly, signal-to-signal interactions are represented implicitly in the present model (for example, IPTG-Lac in Section 4.1.2). The clarity and accuracy of models which include such signal-to-signal interactions could be improved by allowing their explicit representa-

tion [François and Hakim, 2004] through an extension to the symbolic notation (Section 1.7), and by adding corresponding support in the software. Secondly, it would be useful to expand the range of supported types of restriction endonuclease to include those which digest outside their recognition sequence (see Section 4.5). This will require a strategy to cope with the possibility of functional sites, such as operators, being partly mutated (see Section 3.6). Finally, it is possible that a site in a DNA program, from the simulation database, may be digested as an unwanted side effect of a DNA-arranging operation. It would be convenient if the software is able to identify such potential problem locations automatically and, where possible, suggest a base pair mutation that could be applied with minimal predicted effect to the operation of the site. When composing candidate DNA programs, the number of mutations that would need to be applied to existing sites held in the database could also be used as a scoring heuristic.

6.3.3 Quantitative and Standardised Data

The natural biological cell hosts an environment filled with complex interactions and stochasticity. It does not readily lend itself to the standardised measurements of DNA programs and their components, as its evolved purpose is to carry out the processes of life, and not to run synthetic constructs with consistency. Qualitative data used in simulations in this thesis was derived from the literature, as no database with standardised, precise numerical *in vivo* reaction constants for the components was available. There is a need to better understand the workings of biological cells. Considerable effort continues to be spent on unravelling the intricate processes by which a set of genes can drive the development and behaviour of a cell. This knowledge will also increase our understanding of the possibilities and boundaries of DNA programs. However, an alternative approach is to develop a controlled cellular environment of minimal complexity, which may more readily allow analysis. Mushegian and Koonin have proposed a minimal environment of genes for bacterial life through the comparison of common DNA sequences across different bacterial genomes [Mushegian and Koonin, 1996]. Towards realising such a goal, a project has recently been established to sequence the genomes of an organism encoded by under one million nucleotide base pairs, *Mesoplasma florum*, ['Mesoplasma'], and to generate a minimal cellular environment by removing inessential components.

6.3.4 Ciliate Gene Assembly

Another organism that has the potential to further our understanding of rearrangeable DNA programs, is the single-celled eukaryotic *ciliate* [Ehrenfeucht *et al.*, 2004]. Ciliates have two kinds of nucleus: a *micronucleus* that is used for the sexual exchange of DNA and a *macronucleus* that contains genetic information that directs the growth and division of cells [Prescott, 1994]. Macronucleus genes are expressed, whereas genes in the micronuclease are not. When ciliates mate in pairs, the micronucleus is converted into a macronucleus by a process of extensive DNA rearrangement – from hundreds of thousands to several million DNA rearrangements within a few hours [Amos, 2004 *(pages 171-201)*]. This provides a promising indication of a potential rate of DNA program rearrangement that could be attained. The mechanisms that are used by ciliates to rearrange DNA are not fully understood, however, nucleotide markers which appear to direct this process of gene assembly have been identified, called *internal eliminated sequences*. These sequences are removed during the course of DNA rearrangement and they separate genetic material that is to be included in the macronucleus, called *macronuclear destined sequences* (MDSs). The ends of the MDSs contain specific sequences that direct how they are recombined together. A number of models have been presented to describe this process of guided DNA rearrangement based on the excision, insertion and inversion of DNA sequences [Kari and Landweber, 1999], [Prescott *et al.*, 2001]. It would firstly be interesting to model the process of ciliate DNA rearrangement in a single cell using the DNA program software. One could then extend such a model, by allowing for a *population* of ciliates that can mate with each other. Given this, one could investigate the nature of any selective advantages that ciliates may gain through their method of gene assembly.

6.3.5 Biological Scalability using Homing Endonucleases

The DNA-arranging properties of enzymes from the class of restriction endonucleases were studied in this thesis owing to the characterisation and commercial availability of a large number of different types (unlike homing endonuclease genes) and the modularity afforded by cohesive ends of many different enzymes (unlike recombinases). However, for each restriction endonuclease which is expressed by a DNA program, the host cell's DNA must be made immune – for example by methylation, or by the mutation of vulnerable DNA sequences. This is likely to become impractical, or at least cumbersome, when a DNA program can express a range of different

restriction endonucleases, even in a bacterial cell with a genome consisting of a few million nucleotide base pairs. The use of homing endonuclease genes can easily mitigate this problem through their considerably higher specificities. Furthermore, they can even be expressed in eukaryotic cells. However, unlike restriction endonucleases, and despite a growing catalogue of characterised homing endonuclease genes [Roberts *et al.*, 2003], there are presently few examples of groups of enzymes supporting *modularity*: where two or more enzymes recognise different sequences which are able to produce compatible cohesive ends. For example the homing endonucleases, I-*Dmo*I and H-*Dre*I, both produce the same cohesive end of 5'-GTAA-3'; however, they recognise long and different sequences: 5'-ATGCCTTGCCGGGTAAGTTCCGGCGCGCAT-3' and 5'-CAAAACGTCGTAAGTTCCGGCGCG-3', respectively. The cloning and sequencing of existing homing endonucleases will add to the repertoire of potential DNA-arranging enzymes, as will evolutionary protein techniques [Lucas *et al.*, 2001]. This process is, in fact, already underway, in part driven as the properties of homing endonucleases are being used as research tools [Burt, 2003], [Gong and Golic, 2003], [Chevalier *et al.*, 2002], [Jasin, 1996].

Summary

This thesis has presented the concept of *in vivo* computational programs that are encoded in DNA and which can be rearranged in a controlled manner. A symbolic notation was developed to represent DNA programs that are able to rearrange themselves and software was created both to investigate the search space of such DNA programs, and to simulate the emergent behaviours of the stochastic, dynamic genetic networks arising from these DNA programs. One DNA program found by the software, a switch that stores its state in DNA, rather than in protein concentrations, was implemented in wetware. Looking to the future, now that the plausibility of the building blocks of a reprogrammable DNA mechanism has been demonstrated, it will be instructive to integrate these components in wetware to investigate potential *in vivo* DNA storage and *in vivo* DNA functional unit library mechanisms.

Appendix A

Software

This chapter provides further details on how to use the simulation software and how it works. Section A.1 presents the script language for experiments using the DNA Program Generator. Section A.2 describes provisions made in BioSim, the underlying simulation platform, to support modularity and efficiency. Section A.3 presents the settings that are used in the simulation of DNA programs. Section A.4 describes how the search tool positions operator sites when it is generating DNA candidates. Section A.5 formally describes the syntax of the site database file. Section A.6 presents a graphical front-end for the search tool and describes its operation. The software, examples, results, source code and related documents are available on-line at "http://blenkiron.com/phd".

A.1 Search Tool Reference

This section is a software reference guide for the commands used in search criteria read by the DNA Program Generator. An overview of the search and simulation software can be found in Section 3.3. The strategy of the search tool, with example command scripts, is also described in Section 3.4.

A.1.1 Overview

Command *scripts* are composed of three types of statement (Figure A.1.1): *constraints* place limits on the way in which the search space in traversed, *settings* configure search parameters and *trials* describe experimental conditions (*command* statements), and how experiments are assessed (*evaluate* statements).

```
script      ← { constraint | setting }* trial+
trial       ← "trial:" label { command | evaluate }+
command     ← "send" { [units] signal | [units] signals | "reset at" time }
evaluate    ← "check" { expression | sequences | resources } [value]
constraint  ← { include | exclude | specify }
setting     ← "setting:"
                 { "default promoter is" promoter } |
                 { "default operator is" operator } |
                 { "default terminator is" terminator } |
                 { "plasmid backbone is" plasmid } |
                 { { "maximum" | "minimum" } "number of proteins is" value } |
                 { { "maximum" | "minimum" } "number of operators is" value } |
                 { { "maximum" | "minimum" } "number of cut sites is" value } |
                 { "maximum pool size is" value } |
                 { "prune" { "none" | "worst" | "suboptimal" } "layouts" }
                 { "threshold" { "slightly" | "greatly" } "is" value } |
                 { "optimise by" { "length" | "elements" } } |
                 { "choose cut sites by" { "step" | "part" | "gene" | "pair" } } |
                 { "perform" value "runs per trial" }
expression  ← { signal binary { signal | value } } | { signals unary }
sequences   ← "dna at" time "with" protein "is" { "circular" | "linear" }
resources   ← "cost at" time
binary      ← "is" { ["much"] { "less" | "more" } "than" } |
                   { { "less" | "more" } "than" ["or equal to"] } |
                   { ["approximately"] "equal to" }
unary       ← "is" { "increasing" | "decreasing" | "stable" }
include     ← "always use" { pro-list | op-list }
exclude     ← "never use" { pro-list | op-list }
specify     ← "only consider using" { pro-list | op-list }
units       ← value "units of"
signal      ← protein "at" time
signals     ← protein "from" range
range       ← time "to" time
pro-list    ← { "protein" protein } | { "proteins from" protein+ }
op-list     ← { "operator" operator } | { "operators from" operator+ }
label       ← {a..z|A..Z|" "|'|0..9}+
time        ← [0..9] 0..9 ":" 0..5 0..9
value       ← [0..9] 0..9 [ "." 0..9 [ 0..9 ] ]
```

Table A.1.1: The core syntax of the search tool criteria grammar in Extended Backus-Naur Form (EBNF). Each **protein**, **promoter**, **operator**, **terminator** and **plasmid** must correspond to a site identifier in a DNA site database supplied to the search tool. These sites are printed in blue. Symbols are printed in green when they are used (on the right side of a rule).

A.1.2 Constraints

The purpose of placing constraints on a script is to affect the contents of the generated protein and operator sets that are consulted during the construction of DNA programs by the search tool (Sections 3.4.1 and 3.4.2). This can be done to influence

the properties of desired solutions and to increase efficiency by excluding candidates with known, unwanted properties. The three types of constraint command, described next, apply separately to both proteins and operators.

A.1.2.1 include

An include command forces the search tool to always use the sites stated in candidate DNA programs. One or more sites can be used with each include command. For example, `"always use proteins from cro, cI and cII"`, will ensure that the search tool always inserts all three protein encodings into DNA programs. Optimisation (Section 3.7.5) may still remove unnecessary sites that were specified with an include command.

A.1.2.2 specify

A specify command restricts the optional sites considered by the search tool to a supplied subset. For example, `"only consider using operators from oR1 and oR2"`, will prevent the search tool from using operator sites other than these two. However, if the search tool determines that a site is critical to the successful operation of a candidate DNA program, or if an include command uses it, these will take priority over specify commands.

A.1.2.3 exclude

An exclude command prevents the search tool from using a site in a DNA program. For example, `"never use protein bfp"`, will cause the search tool to avoid using this protein. Sites that the search tool calculates are critical to the operation of a DNA program candidate and sites used in include commands will override parts of any exclude command they conflict with.

A.1.3 Settings

Settings affect the configuration and modes of operation of the search tool. The following sections describe the categories of setting command.

A.1.3.1 default

The default commands specify the promoter, operator and terminator sites that the search tool should first try to include in DNA program candidates when it encounters a choice. For example, `"default promoter is pT7"`, will make P_{T7} the default promoter.

A.1.3.2 plasmid backbone

This command allows the plasmid backbone of DNA programs to be specified. For example, `"plasmid backbone is pUC18"`.

A.1.3.3 maximum/minimum number

These commands place upper and lower numerical bounds on the number of protein, operator and restriction sites allowed in DNA program candidates. For example, `"maximum number of proteins is 3"`, limits the search tool to using at most three protein sites in each DNA program candidate. By default, the maximum number of protein and operator sites is four.

A.1.3.4 maximum pool size

The maximum number of DNA program candidates that can be considered by the search tool simultaneously can be specified by this command. For example, `"maximum pool size is 100"`. If the number of pool candidates under consideration exceeds the specified limit, then lower-scoring candidates will be removed from the pool and replaced by higher-scoring candidates. By default, the maximum pool size is 10.

A.1.3.5 runs per trial

Simulation runs can be performed more than once using different pseudo-random seeds to aid the assessment of the behavioural variation of DNA program candidates caused by stochastic events. This command specifies the number of runs to perform, e.g., `"perform 10 runs per trial"`. By default, three runs are carried out per trial.

A.1.3.6 prune

There are three methods of pruning candidates before they are simulated (Section 3.5.2), e.g., `"prune suboptimal layouts"`, will discard all generated DNA program

candidates before simulation which cannot obtain the maximum score allowed by the criteria. By default, sub-optimal layouts are pruned.

A.1.3.7 threshold

Two thresholds, `slightly` and `greatly` are used by the search tool to predict the maximum scores of candidate DNA programs (Section 3.5) and to evaluate their simulation results (Section 3.7.4). These thresholds can be set in an experiment's criteria: e.g., `"threshold slightly is 0.05"`, sets the `slightly` threshold to 5%. By default, `slightly` is set to 10% and `greatly` is set to 50%.

A.1.3.8 optimise

Some candidate DNA programs may achieve the same score, based on how well they meet the criteria. A secondary method of evaluation can consider either a candidate length (`length`), in base pairs, or the number of sites required to compose the candidate (`elements`), e.g., `"optimise by elements"`. By default, `length` mode is used.

A.1.3.9 choose restriction sites

Once a DNA program candidate's promoter, operator, protein and terminator sites have been arranged, restriction sites may be placed between them (Section 3.6). As defined in Section 3.6.2, the step size, σ, and distance, δ, are provided with four configurations: `step` sets $\{\sigma = 1, \delta = 1\}$; `part` sets $\{\sigma = 2, \delta = 1\}$; `gene` sets $\{\sigma = 4, \delta = 1\}$ and `pair` sets $\{\sigma = 4, \delta = 2\}$, e.g., `"chose cut sites by gene"`. By default, `part` mode is used.

A.1.4 Trials

A trial describes environmental conditions and rules for evaluating how well a DNA program candidate performs. Protein *signals* can be sent into the simulation environment at time points and across intervals. An experiment script can contain one or more trials and the scores attained from meeting rules in each trial are accumulated to form an overall score for each candidate. The performance of a candidate can be assessed with: comparative *expressions* over two protein assays at a specified time point (binary expression), the change in a series of assays of the same protein over a time period (unary expression), the structure of the DNA (sequences expression) and the cost of protein synthesis (resources expression).

A.1.4.1 signal

The amount of a protein (Section A.3.6) can be increased by sending protein signals, e.g., `"send 0.1 units of IPTG at 12:00"`. It is also possible to remove proteins from the simulation by sending a *reset* command, e.g., `"send reset at 24:00"`.

A.1.4.2 binary

Binary evaluation rules compare two assayed values, or an assayed value with a specified value, e.g., `"check gfp at 0:30 is less than bfp at 0:30"`, compares the amount of two proteins (Section A.3.6) at the same time point and, `"check cII at 1:20 is approximately 0"`, compares the amount of a protein against a value. The list of available binary operators can be found in Table A.1.4.2. A description of the proportions used in the table can be found in Section A.1.3.7.

Rule Template	Evaluation
x `is less than` y	$x < y$
x `is more than` y	$x > y$
x `is much less than` y	$x(1+G) < y$
x `is much more than` y	$x > y(1+G)$
x `is less than or equal to` y	$x \leq y$
x `is more than or equal to` y	$x \geq y$
x `is equal to` y	$x \leq y \wedge x \leq y$
x `is approximately equal to` y	$x \leq y(1+S) \wedge x(1+S) \leq y$ $\vee\, x+S \leq y \wedge x \leq y+S$

Table A.1.4.2: The set of binary operators, over protein assays and values, which can be used in rules. These operators apply to single time points.

A.1.4.3 unary

Unary evaluation rules compare the change in the assay of a protein over time. For example, `"check cre from 1:00 to 2:00 is increasing"`. The list of available unary operators can be found in Table A.1.4.3. A description of the proportions used in the table can be found in Section A.1.3.7.

Rule Template	Evaluation
x `is increasing from` t_0 `to` t_1	$x^{t_0}(1+G) < x^{t_1}$
x `is decreasing from` t_0 `to` t_1	$x^{t_1}(1+G) < x^{t_0}$
x `is stable from` t_0 `to` t_1	$x^{t_0}(1+G) > x^{t_1}$ $\wedge x^{t_1}(1+G) > x^{t_0}$

Table A.1.4.3: The set of unary operators, over protein assays, which can be used in rules. These operators apply to a time interval.

A.1.4.4 sequences

The sequence rule allows the DNA in a simulation run, specific to a particular site, to be tested for whether it is linear or circular (Section 1.5.5). For example, `"check DNA at 48:00 with ori is circular"`.

A.1.4.5 resources

An indication of the cost of running a DNA program candidate by a simulated host cell is obtained by counting the number of base pairs synthesised during the translation of mRNA templates. This cost can be used in evaluation rules, e.g., `"check cost at time 0:10 is less than 1,000,000"`.

A.1.4.6 weight

Rules can be modified with a weight value to stress the relative importance of each rule. A five-point scale is shown in Table A.1.4.6, e.g., `"check cII at 1:00 is much more than 0 (very important)"`.

Rule Template	Weight
`not important`	1
`quite important`	2
`important`	4
`very important`	8
`extremely important`	16

Table A.1.4.6: Weighting values that can be optionally applied to rules. The default weight is `important`. The weights of rules in a trial which evaluate to be true are summed together to produce a score for each DNA program candidate.

A.2 BiSim Modularity and Efficiency

This section describes provisions made, in BiSim, the simulation platform, to assist the process of designing and experimenting with simulation models.

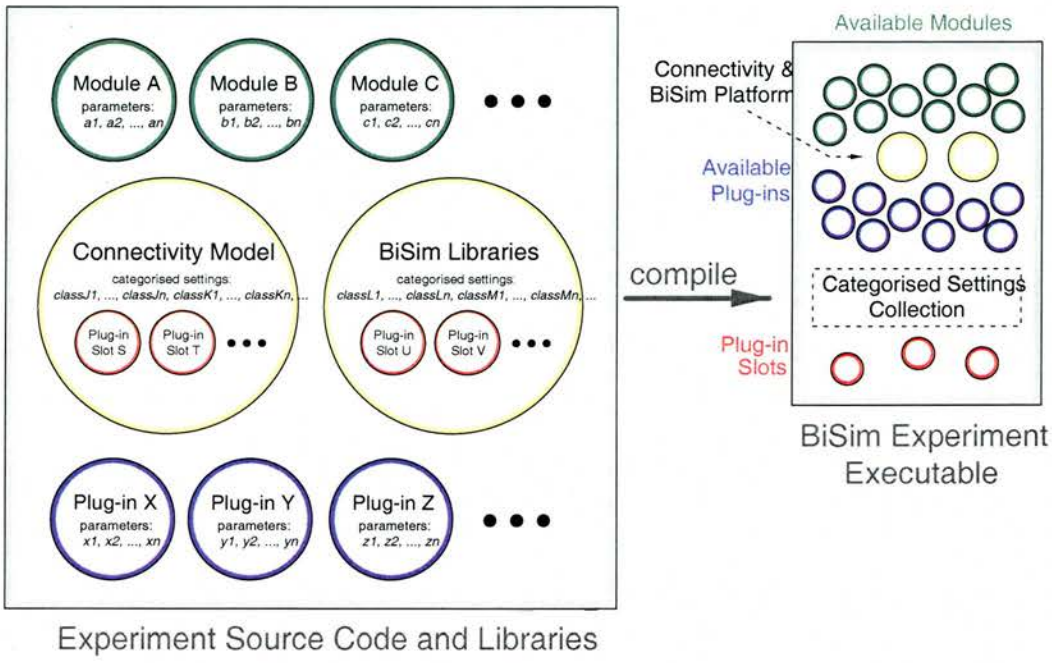


Figure A.2a: The composition of a BiSim executable. An executable is composed of the BiSim program libraries, a collection of modules conferring behavioural properties to nodes (e.g. molecule attachment site), a connectivity model describing the graphs which connect nodes (e.g. DNA helix) and a collection of plug-ins which define properties of messages passed over the graphs (e.g. by diffusion).

To aid efficiency, for each set of simulation experiments, a stand-alone executable is created by combining experiment-specific code with BiSim's libraries and sample modules (Figure A.2a). This executable can be configured using an automatically generated script file for a particular experiment (Figure A.2b). The script file holds parameters, communication settings, called *plugins*, and device composition specifications. Device compositions are compiled into a virtual machine from their constituent modules, their parameters and connections *on-the-fly*. This is convenient when experimenting with alternative device models.

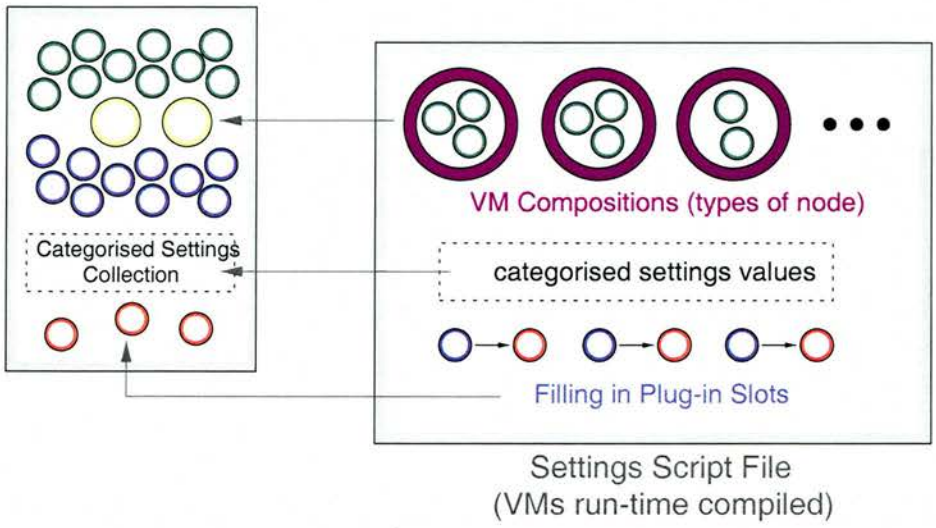


Figure A.2b: The composition of a BiSim settings script file. A settings script file declares different types of virtual machine node (e.g. operator site) by composing collections of modules and their parameters. Plug-in slots are filled in for the chosen connectivity model to specify properties of the model's graphs. Numerical and textual parameters can also be set.

Programs that run on devices can be read from an external program file and are also compiled *on-the-fly* against a compiled virtual machine to mitigate the efficiency cost of this modularity (Figure A.2c). More detail can be found in the BiSim *User Guide* [Blenkiron, 2004].

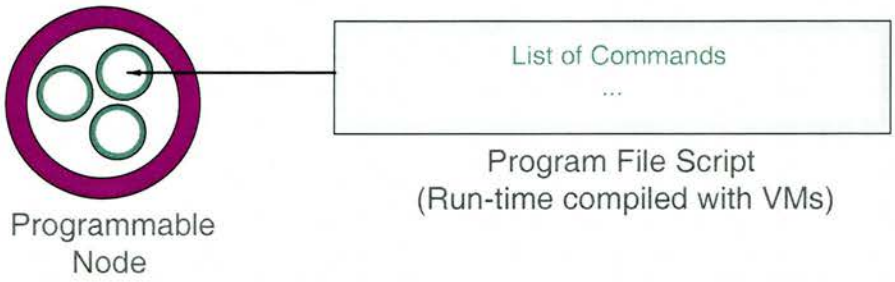


Figure A.2c: The composition of a BioSim node program. Programs, which are loaded by nodes to determine their behaviour, can be specified using any of the functionalities that the modules composing the nodes confer.

A.3 Simulation Parameters and Units

This section describes the settings that can be used to configure how DNA program candidates are simulated and the meaning of the units used.

A.3.1 `stickiness`

The relative bond strengths between different cohesive ends can be specified by a list of ssDNA bases, using standard abbreviations to represent ambiguity, followed by a percentage. For example, `"stickiness CCAG=42%,CTAG=40%"`.

A.3.2 `circular`

This parameter specifies how more likely a single strand of DNA with compatible ends is to ligate to itself, rather than to a second, similar piece of DNA.

A.3.3 `labels`

A list of proteins can be tagged with a colour which will be used in graphical output. For example, `"labels gfp=green,rnap=cyan"`.

A.3.4 `read_speed`

The number of codons, per unit time (`0:01`), that RNA polymerase is able to transcribe.

A.3.5 `mRNA unit`

A unit of mRNA is defined as: the amount of mRNA produced when RNA polymerase transcribes a gene.

A.3.6 `protein unit`

A unit of simulated protein is defined as: the amount of protein produced when a simulated unit of mRNA is translated.

A.3.7 `proximity`

This parameter determines, per unit time (`0:01`), per unit of ligase, the proportion of time the ligase is sufficiently close to DNA to be able to initiate the rejoining of two nearby, suitably positioned cohesive ends.

A.4 Operator Layout Algorithm

This section presents pseudo-code of ω, the operator layout algorithm, used in Section 3.4.4. Its purpose is to generate arrangements of up to n operator sites (with two orientations each) in n slots (insertion points) such that in at least one arrangement: any two operator sites are adjacent with (a) the same (b) different orientations, that any operator site is positioned in the first slot[1], and that there are no more than $2n$ arrangements generated (Figure A.4).

[1]The first slot is special as this is adjacent to one gene only.

```
1  pairs = ∅
2  for 0 ≤ i < |X|:
3     orientation = "←"
4     route^A = { i→ }
5     route^B = { i^orientation }
6     repeat until |route| = |X|
7        let k = value of route^A_0
8        find the min j if it exists where: j > k ∧ (k, j) ∉ pairs
9        otherwise, find the min j where: (k, j) ∉ pairs
A        invert orientation
B        prepend j→ to route^A and prepend j^orientation to route^B
C        add (k, j) to pairs
D     ω_2i ⇐ {0, ..., |X| − 1} ↦ route^A,  ω_2i+1 ⇐ {0, ..., |X| − 1} ↦ route^B
```

Figure A.4: Pseudo-code of operator site arrangement algorithm, ω. The set, **X**, contains protein sites and both the maximum number of operator sites and slots, n, is set to $|\mathbf{X}|$. The algorithm ensures that each ordered pair of operator site indices $(j,k) : j \neq k,\ 0 \leq j,k < |\boldsymbol{X}|$ will be next to each other in at least one of the arrangements generated (lines 7..9). Also, within the generated arrangements, each site is placed in the first slot (lines 2,4,5) and adjacent sites have both the same and different orientations (lines 3,A,B).

A.5 DNA Site Database Format

This section describes the format of the database of DNA sites and their properties which is used by the search tool to generate DNA program candidates (Section 3.3). Databases hold information on plasmids, protein encodings, promoters, operators, terminators and the effects of DNA-arranging enzymes. A range of genetic sites with well-studied qualitative properties were included in the database. Reaction constants were obtained by numerical approximations of descriptions of cellular interactions in the literature. Specific details are included with the software distribution which is available on-line at "http://blenkiron.com/phd/". Figure A.5 specifies the syntax of a database file.

```
database         ← { plasmid- | protein- | promoter- | operator- | terminator-
                     | restrict- }* data
plasmid-data     ← "> plasmid" plasmid sequence
protein-data     ← "> protein" protein "decay(" value ")" sequence
promoter-data    ← "> promoter" promoter "affinity(" value ")" sequence
operator-data    ← "> operator" terminator { { "> up(" | "> down(" }
                     protein, value, value, promoter, value ")" }+ sequence
terminator-data  ← "> terminator" restrict "effective(" value ")" sequence
restrict-data    ← "> restrict" protein pattern offset offset value
sequence         ← { A | T | G | C }+
pattern          ← { A | T | G | C | R | Y | M | K | S | W | B | D | H | V | N }+
site-label       ← { a..z | A..Z | 0..9 }
value            ← [0..9] 0..9 [ "." 0..9 [ 0..9 ] ]
offset           ← { + | - } [0..9] 0..9
```

Table A.5: The syntax of the DNA site database grammar in Extended Backus-Naur Form (EBNF). Each definition of a site must begin on a new line and may extend over multiple lines. Nucleotide patterns use the standard abbreviations for ambiguity (Section 1.5.6). Each protein, promoter, operator, terminator and plasmid is identified by a unique, alphanumeric name and is printed in blue. Symbols are printed in green when they are used (on the right side of a rule).

A.6 Search Tool Graphical Interface

This section presents the graphical user interface of the DNA Program Generator (Section 3.4). This search tool's interface can be used in two ways: it can be given a set of candidate criteria (Section A.1) and it will attempt to generate acceptable DNA program candidates, and it can also simulate a specific candidate which was either found during an earlier search or composed by hand. The following sections describe the elements of the graphical interface shown in Figure A.6.

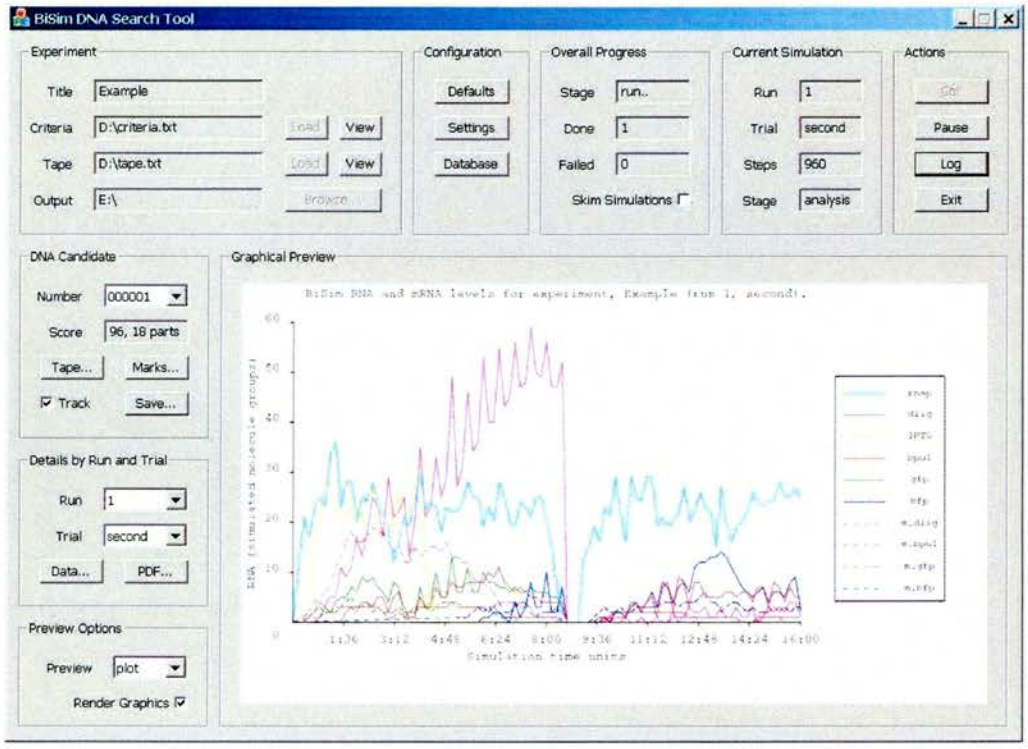


Figure A.6: A screen-shot of the DNA Program Generator's graphical user interface. This interface runs in a Microsoft Windows™ environment. However, the core search and simulation code is platform independent.

A.6.1 Experiment

The *experiment* group allows the user to select and view experiment files, to specify where to save search results and temporary files, and to provide the title of an experiment.

A.6.1.1 Title

The *title* of a search experiment is prefixed to the name of all results files for ease of identification when running several experiments in sequence. It is also is used inside graphical outputs and data files.

A.6.1.2 Criteria

A *criteria* file (Section A.1), which specifies the environmental conditions of experiments and provides a means to evaluate the utility of candidates, can be selected using the *load* button. The *view* button allows a loaded criteria file to be inspected and modified.

A.6.1.3 Tape

Optionally, a specific DNA program candidate (*tape*) can be specified (using the *load* button) rather than allowing the search tool to generate candidates. The *view* button allows a loaded tape file to be inspected and modified.

A.6.1.4 Output

The *output* directory can be specified using the *browse* button. All temporary files created by the search tool, simulation and graphical software will be placed here and removed when the graphical user interface is closed. This directory will also be used as the default location for saving results files.

A.6.2 Configuration

The *configuration* group allows software parameters to be set and the database of DNA sites and properties to be selected.

A.6.2.1 Defaults

This button is useful when modifying or creating a new simulation model. It configures the BioSim simulation platform parameters which are used for debugging and optimisation information.

A.6.2.2 Settings

This button allows the simulation model's parameters and units, which were described in Section A.3, to be inspected and modified.

A.6.2.3 Database

The *database* of DNA sites and their properties, described in Section A.5, can be inspected and modified using this button.

A.6.3 Actions

The *actions* group controls the running of the search tool software.

A.6.3.1 Go

Once a criteria file has been loaded (Section A.6.1.2), pressing the *go* button will cause the search tool to start generating DNA program candidates if no tape (Section A.6.1.3) has been specified. The search tool will continue running until it is unable to improve the scores of the candidates held in its pool (Section A.1.3.4). The *overall progress* of a search is shown in another group (Section A.6.4). When a tape has been specified, the search tool will evaluate only its performance with the criteria.

After the tool has finished searching, an option is presented to the user to optimise (Section 3.7.5) the highest scoring candidates, in the pool. The optimisation process will continue until none of the candidates in the pool can be improved.

A.6.3.2 Pause

The search tool will *pause* and stop generating DNA program candidates. If a simulation run is in progress, this will be completed first.

A.6.3.3 Log

This button displays a log of the decisions made by the search tool with respect to the composition of site sets and stages described in Section 3.4.

A.6.3.4 Exit

This button closes the search tool. If the results of an experiment have not been saved, the tool will prompt the user before closing.

A.6.4 Overall Progress

The *overall progress* group shows the state of a search process. It also allows the search tool to quickly estimate the search space size by generating candidates, but skipping simulations.

A.6.4.1 Stage

This control shows the main stage the search tool is in: *loading* when experiment files are being loaded, *search* when the search tool is generating candidates, *run* if a specified tape is being simulated, *pause* when the software has been paused, *optimise* during the optimisation process, *done* when optimisation is complete and *closing* when the software is tidying up temporary files before is closes down.

A.6.4.2 Done

This control counts the number of DNA candidate programs, which have not been pruned (Section 3.5) and have been successfully simulated and evaluated with the criteria.

A.6.4.3 Failed

This control shows how many DNA candidate programs could not be simulated in one of their runs. An experiment may fail because limits of the simulation model have been exceeded (for example, digestion of the end of a protein encoding site is not currently supported), or because of a software error (for example, running out of storage space for files in the temporary directory). Any DNA candidate programs that cause a failure are saved into enumerated files in the output directory with a ".failed" extension. These files are not cleared when the search tool closes.

A.6.4.4 Skim Simulations

Sometimes it is useful to estimate the size of the search space before starting an experiment. For example, this can provide an indication of how long an experiment will take to complete. The *skim simulations* button instructs the search tool to generate DNA program candidates as normal, but to skip their simulation. The *done* control (Section A.6.4.2) will count the number of candidates generated. The estimate provided will be an upper bound, as the *actual* scores of previously-simulated candidates can be used to help prune newly-generated candidates with lower predicted scores (Section 3.5).

A.6.5 Current Simulation

The *current simulation* group provides details of the simulation progress of the current DNA program candidate which the search tool is considering.

A.6.5.1 Run

This control shows the current *run* of an experiment (Section A.1.3.5). Each run provides the random number generator with a different seed.

A.6.5.2 Trial

The label of the current trial, as specified in Section A.1.4, is displayed by this control.

A.6.5.3 Steps

This control shows the number of simulated time units required to complete the current run's simulation.

A.6.5.4 Stage

Initially when a DNA program candidate is simulated, this is performed in *scoring* mode. This mode simulates the candidate to obtain a table of protein assays and DNA sequence information to allow the candidate's evaluation with a set of criteria.

When a candidate's score is sufficiently high that it is inserted into the pool of candidates held by the search tool, it will undergo a second round of simulation, under identical conditions, in *analysis* mode. In this mode, more detailed information is produced by the simulation model to justify how a candidate obtained its score.

A.6.6 DNA Candidate

The DNA candidate group holds a pool of the best generated programs during the course of a search. It also maintains justifications for the overall scores of the candidates in the pool.

A.6.6.1 Number

Each DNA program candidate which is inserted into the pool, is allocated a number from 1 upwards. This list box allows a candidate to be selected which will affect the display and operation of other controls which are specific to the current candidate.

A.6.6.2 Score

The score of the selected pool candidate is shown in two parts. The first part is the sum of all of the weights of the criteria rules that it has passed. The second part is an indication of the size of the candidate measured either in base pairs or the number of sites required (Section A.1.3.8).

A.6.6.3 Tape

This button will launch a window displaying the selected pool candidate's layout in the notation presented in Chapter 1.7.

A.6.6.4 Marks

Pressing the *marks* button for a selected pool candidate will show how the candidate obtained its score. The runs in which each of the rules was passed are laid out for each trial in the criteria.

A.6.6.5 Track

Whenever a new candidate is generated, and when tracking is turned on, this candidate will be automatically selected and cause all of the other dependent controls to reflect its information.

A.6.6.6 Save

Pressing the *save* button will prompt the user to select a directory into which a collection of results for a candidate is saved. Results are composed of the tape layout, justification for its score, and, for each trial and run, both a table of protein assays and a graphical illustration of the simulation experiment (Section A.6.7.4).

A.6.7 Details by Run and Trial

This group allows a specific simulated run and trial of the currently selected pool candidate to be chosen and analysed.

A.6.7.1 Run

This list box selects one of the simulation runs of the candidate.

A.6.7.2 Trial

A *trial*, for a selected run, can be chosen using this list box.

A.6.7.3 Data

This button will launch the table of protein assays for the selected pool candidate, run and trial.

A.6.7.4 PDF

Illustrations of DNA structure, protein attachments and a graph of protein assays during the course of a simulation, are compiled into a booklet after each simulation run. Pressing the *PDF* button will show the booklet which is provided in the Adobe Acrobat™ Portable Document Format.

A.6.8 Preview Options

While the search tool is generating and simulating candidates, the graphical interface can display an illustration of the selected DNA program candidate and a graph of its protein assays. When candidate tracking is on (Section A.6.6.5), the most recently simulated DNA program candidate will be displayed.

A.6.8.1 Preview

This list box selects whether a graph of protein assays (*plot*) or the first time point (`0:00`) of the simulation (*intro*) is displayed inside the *Graphical Preview* group (Section A.6.9).

A.6.8.2 Render Graphics

By default, the search tool will generate simulation illustration booklets and graphical previews for every DNA program candidate that is inserted into the candidate pool. The DNA Program Simulator's graphical rendering is considerably more time consuming than its simulation. Turning this check box off will prevent the simulator from generating graphics. However, it is possible to save the tape of a generated candidate (Section A.6.6.6), which was generated without graphics, and then to render its graphics at a later time (Section A.6.1.3).

A.6.9 Graphical Preview

This group contains a graphical preview of the currently selected pool candidate. A more detailed and higher quality graphical illustration can be obtained by viewing the associated PDF file (A.6.7.4).

Appendix B

Supplementary Information

This chapter contains supplementary information relating to the wetware experiments performed in Chapter 5. The first section describes details of the biological protocols that were used in experiments. The subsequent section presents typical photographs from my lab books that were used to verify the intermediate steps of experiments. The third section contains the annotated nucleotide base sequences of the target plasmids that were constructed for biological experiments.

B.1 Biological Solutions and Techniques

This section describes the solutions, techniques and equipment that were used during experiments. All biological manipulations were carried out as described here, unless explicitly stated otherwise.

B.1.1 In Vitro Solutions

B.1.1.1 dNTPs

Stock 10mM dNTPs solution was made by adding distilled water to the four constituent d*N*TPs (Sigma) as shown in Table B.1.1.1. Working stocks were stored at -20°C in 25$\mu\ell$ aliquots.

Table B.1.1.1: dNTPs solution.

dATP, 100mM	25$\mu\ell$
dCTP, 100mM	25$\mu\ell$
dGTP, 100mM	25$\mu\ell$
dTTP, 100mM	25$\mu\ell$
Distilled water	150$\mu\ell$
dNTPs, 10mM	250$\mu\ell$

B.1.1.2 PCR Buffer

Stock PCR buffer was mixed as shown in Table B.1.1.2. 25$\mu\ell$ aliquots were stored at -20°C. When preparing a solution for a specific PCR reaction, 22$\mu\ell$ of stock PCR buffer was made up to a volume of 25$\mu\ell$ with 0.5$\mu\ell$ Taq DNA Polymerase (Promega), DNA, primers and distilled water.

Table B.1.1.2: PCR Buffer.

Mg Free Buffer, 10x (Promega)	50$\mu\ell$
$MgCl_2$, 15mM (Promega)	30$\mu\ell$
dNTPs, 10mM (Stock)	10$\mu\ell$
Distilled water	350$\mu\ell$
PCR Buffer	440$\mu\ell$

B.1.1.3 Oligonucleotide Synthesis

All primers and other oligonucleotides were synthesised by MWG Biotech. The software, *Primer3* [Rozen and Skaletsky, 2000], was used to help choose suitable primers. Stock solutions were made by dilution to a concentration of 100ng/$\mu\ell$ and were stored at -20°C.

B.1.1.4 TBE Buffer (10x)

Stock 10x TBE buffer was made as shown in Table B.1.1.4 and was stored at room temperature. Stock solution was diluted to 1x concentration with distilled water prior to use in gel electrophoresis.

Table B.1.1.4: TBE Buffer.

Tris base (Sigma)	108g
Boric acid (Sigma)	55g
EDTA (Sigma)	9.3g
Distilled water	up to 1ℓ
TBE Buffer, 10x	1 ℓ

B.1.2 In Vitro Techniques

B.1.2.1 Gel Electrophoresis

Gel electrophoresis was used to determine the size and presence of strands of DNA samples and to isolate DNA of a certain size from a mixture of different sizes. Agarose gels were made by dissolving an amount of electrophoresis grade agarose (Invitrogen), appropriate to the expected size of the DNA being run, in 1x TBE stock solution by heating. 1$\mu\ell$ ethidium bromide was added to the agarose solution before pouring it into a 8100 series horizontal mini-gel tank (Molecular Bio-Products) and leaving it to set at 4°C. DNA samples were run in 1x TBE stock solution. When running DNA samples of several kilo base pairs overnight, additional ethidium bromide was added to the solution, if necessary.

100bp and 200bp markers (Promega) were used to estimate the size and amount of DNA in the samples. Gels were viewed over an IBI transilluminator UV source and an IBI EP H-S Polaroid Hood was used to capture an image of each run.

B.1.2.2 DNA Sequencing

All DNA sequencing was carried out by DBS Genomics, University of Durham. An estimated $2\mu g$ of DNA (by comparison with DNA markers) per sample was sent in $20\mu\ell$ of distilled water after purification with an appropriate kit described in Section B.1.2.6.

B.1.2.3 DNA Primers

Table B.1.2.3 summarises the primers used to amplify, detect and sequence DNA samples. A subscript of *L* (left) indicates a primer that attaches to an anti-sense strand. Conversely, a subscript of *R* (right) attaches to a sense strand.

Table B.1.2.3: Primer Sequences.

5'-GAGGCCCTTCCTAGGAGGGAA-3' ($long_L$)
5'-CAACAGATGGCTAGCCATATG-3' ($long_R$)
5'-GCTCGTTTAGTGAACCGTCAG-3' (gfp_L)
5'-CAACAGATGGCTGGCAACTA-3' (gfp_R)
5'-CTTGGTTATGCCGGTACTGC-3' (bfp_L)
5'-GGTGATGTCGGCGATATAGG-3' (bfp_R)
5'-ATTTAGGTGACACTATAG-3' ($sp6$)
5'-TTGTAAAACGACGGCCAGTG-3' ($M13_L$)
5'-CACACAGGAAACAGCTATGACC-3' ($M13_R$)

B.1.2.4 Digestion of DNA

Digestion of DNA with restriction endonucleases was carried out in accordance with the relevant manufacturer's guidelines. When double-digests were performed, conditions presented in the Fermentas *Five Buffer Plus System* chart were used. The conditions of digestions during "rotation" experiments (Section 5.2) often deviated from manufacturer's guidelines in order to optimise their yield.

B.1.2.5 Ligations

Ligations of DNA into vectors were carried out using reagents and guidelines from the *pGEM-T Easy Vector System* (Promega), including the *pUC18* vector (Fermentas). Reactions were carried out at 4°C overnight. Stubborn ligations which produced a low yield were performed with $5u/\mu\ell$ T4 DNA Ligase with PEG (Fermentas). Some "rotation" experiments (Section 5.2) deviated from manufacturer guidelines in order

to discourage star-activity of the *Bpu*10I restriction endonuclease while allowing T4 DNA ligase to work effectively.

B.1.2.6 DNA Purification

All purification steps were performed using kits from Qiagen. The *PCR Purification Kit* was used, in addition to PCR cleanup, for routine enzymatic cleanup. *Qiaex II Gel Extraction Kit* was used to recover DNA from agarose gels. The *QIAprep Spin Plasmid Kit* (mini-prep) was used for plasmid extraction from colonies.

B.1.2.7 Fluorimeter Measurements

Fluorimeter measurements were made with a SPEX FluoroMax 5030 (Instruments SA). $0.5\mu\ell$ of sample was diluted immediately before use with distilled water to up to $2\mu\ell$ in a cuvette (Optiglass, catalogue number 9002). Measurements were taken in 1nm increments and averaged over three readings. A control experiment was performed for each set samples with a solution of JM109-DE3 (Promega) cells, without vectors, having a similar optical density to the samples. The background measurements of control readings was subtracted from the sample measurements.

B.1.2.8 PCR

Hot-start, touch down PCR was performed for all PCR reactions using a DNA Engine Opticon machine with the cycles shown in Table B.1.2.8.

Table B.1.2.8: PCR Cycles.

temperature	period	cycles
94°C	5 min	1
94°C	1 min	
(65-α)°C	30s	$\alpha = \{0..6\}$
72°C	2 min	
94°C	1 min	
58°C	30s	20
72°C	2 min	
72°C	10 min	1

B.1.3 In Vivo Solutions

B.1.3.1 Luria-Bertani Broth

Luria-Bertani Broth was prepared as shown in Table B.1.3.1, autoclaved, and then stored at room temperature.

Table B.1.3.1: LB Broth.

Glucose (Sigma)	1.0g
Sodium chloride (Sigma)	10.0g
Tryptone (Sigma)	10.0g
Yeast extract (Sigma)	5.0g
Distilled water	up to 1ℓ
LB Broth	1ℓ

B.1.3.2 Luria-Bertani Agar

Luria-Bertani Agar was made as shown in Table B.1.3.2, autoclaved, and then poured into 85mm plates when its temperature had cooled to 50°C. Any required additives (ampicillin, x-gal and IPTG) were mixed immediately before pouring. Batches of plates were dated, wrapped in cling-film and stored at 4°.

Table B.1.3.2: LB Agar.

Agar (Sigma)	15.0g
Sodium chloride (Sigma)	10.0g
Tryptone (Sigma)	10.0g
Yeast extract (Sigma)	5.0g
Distilled water	up to 1ℓ
LB Agar	1ℓ

B.1.3.3 TFB I

TFB I solution was composed as shown in Table B.1.3.3. The pH was adjusted to 5.8 with 1M acetic acid. The solution was then filter sterilised and stored at room temperature.

Table B.1.3.3: TFB I solution.

Potassium acetate (Sigma) 30mM	1.47g
Calcium chloride (Sigma) 10mM	0.56g
Manganese chloride (Sigma) 50mM	4.95g
Rubidium chloride (Sigma) 100mM	6.05g
15% glycerol (Sigma)	
Distilled water with	up to 0.5ℓ
TFB I	0.5ℓ

B.1.3.4 TFB II

TFB II solution was composed as shown in Table B.1.3.4. The pH was adjusted to 6.5 with 1M KOH. The solution was then filter sterilised and stored at room temperature.

Table B.1.3.4: TFB II solution.

PIPES pH6.8 (Sigma) 100mM	15.12g
Calcium chloride (Sigma) 75mM	4.16g
Rubidium chloride (Sigma) 10mM	0.60g
15% glycerol (Sigma)	
Distilled water with	up to 0.5ℓ
TFB I	0.5ℓ

B.1.3.5 Ampicillin (500x)

Ampicillin solution was made by diluting 50mg D(-)-α-aminobenzyl penicillin sodium salt in 1mℓ ethanol and was stored at -20°.

B.1.3.6 X-gal (500x)

X-Gal solution was made by diluting 40mg 5-bromo-4-chloro-3-indolyl-β-D-galactopyranoside in 2mℓ N,N'-dimethylformamide. The containing tube was wrapped in aluminium foil and stored at -20°.

B.1.3.7 IPTG (0.1M)

IPTG stock solution was made by dissolving 1.2g IPTG in 50mℓ distilled water and filter sterilising. This solution was stored at 4°C.

B.1.4 In Vivo Techniques

B.1.4.1 Competent Cells

Stocks of JM109-DE3 cells for expression were made using a rubidium chloride method (TFB I and II) by following the manufacturer's guidelines (Promega Subcloning Notebook). However, as blue/white screening was unsuitable for these JM109-DE3 cells, the stage of the protocol for selection of cells containing an F' episome was unnecessary and therefore omitted. Competent cell stocks were quick frozen in a dry ice and isopropanol bath and then stored at -70°C.

B.1.4.2 Transfection

Sample DNA plasmid concentration was determined either by estimation with gel electrophoresis, running a linearised plasmid alongside a marker, or by following the vector manufacturer's instructions when performing a standard ligation of an insert into a vector. Approximately 25ng of sample DNA was added to either 50$\mu\ell$ JM109 or JM109-DE3 cells which had been thawed on ice for 15 minutes in a volume of 1 to 5 $\mu\ell$ in 1.5mℓ eppendorf tubes. The suspensions were incubated on ice for 30 minutes and then immersed in a 42°C water bath for exactly 45 seconds, followed by incubation on ice for 2 minutes. LB Broth was then added to the tubes up to 1mℓ and they were then incubated at 37°C in an orbital shaker at 150rpm for 2 hours.

For each transfection, 10$\mu\ell$ and 100$\mu\ell$ of cells were spread on agar plates, containing 1x ampicillin for selection of antibiotic resistance, using a sterile glass rod. When using JM109 cells, plates also contained 1x X-gal to allow for blue/white screening. For each batch of transfections, a control plasmid was also transfected to estimate the transformation efficiency. Plates were incubated at 37°C overnight and then stored at 4°C.

B.1.4.3 Colony Storage

A colony was picked from a plate, using a sterile glass rod, and used to inoculate 5ml Luria-Bertani broth with 1x ampicillin in a 50ml Falcon Tube. The culture was incubated at 37°C overnight at 150rpm in an orbital shaker. The tubes were then centrifuged at 2000rpm, the supernatant was removed, and the cells were resuspended in 680$\mu\ell$ LB Broth with 150$\mu\ell$ 80% glycerol. The suspension was then stored in a 1.0ml cryogenic storage tube (Nunc) at -70°C.

B.1.4.4 Blue/White Screening

When using JM109 cells for cloning, blue/white screening was often used to assist the selection of colonies containing inserts during transfections (Section B.1.4.2). Cells were spread on agar plates containing 1x X-gal. Those colonies containing an insert interrupted a *lacZ* gene in the vector cloning site (*pGEM-T Easy*, Promega and *pUC18*, Fermentas) which either stopped or reduced production of beta-galactosidase, which metabolises X-gal, and in doing so, produces a blue product. Consequently, colonies which were white or lighter blue, were more likely to contain an insert.

B.1.4.5 Extraction of plasmid DNA

A selected colony was picked from a plate using a sterile glass rod and was used to inoculate 5mℓ Luria-Bertani Broth in a 50mℓ Falcon Tube. The picked colony was also spotted onto another plate containing 1x ampicillin. The culture was incubated at 37°C overnight at 150rpm in an orbital shaker. Plasmid DNA was then extracted using the *QIAprep Spin Miniprep Kit* (Qiagen). Gel electrophoresis was then used to estimate the amount of DNA extracted for each colony.

B.2 Sample Gels

Gel electrophoresis (Section B.1.2.1) was used extensively to aid and verify the intermediate steps of wetware experiments. This section presents a selection of Polaroid pictures of gels from three different steps, accompanied by brief descriptions.

B.2.1 PCR

During the first wetware construction stage (Section 5.1.1) a "longmer" was ligated into a *pGEM-T* vector which was then transfected into competent JM109 cells. To verify the presence of the inserts and to determine their orientations from five picked colonies (*A* to *E*), DNA was extracted from each colony by mini-prep and was then subjected to PCR using three primer pairs (Figure B.2.1). The primers $long_L$ and $long_R$ matched each 5' end of the "longmer". The *SP6* primer matched part of the SP6 promoter in the *pGEM-T* vector that was oriented towards the cloning site. Therefore, $long_L$ and $long_R$ (group *1P*) were used to verify the presence of the "longmer", $long_L$ and *SP6* (group *1Q*) were used to detect a "longmer" pointing towards the SP6 promoter and $long_R$ and *SP6* (group *1R*) were used to detect a "longmer" that had been inserted downstream of the *SP6* promoter.

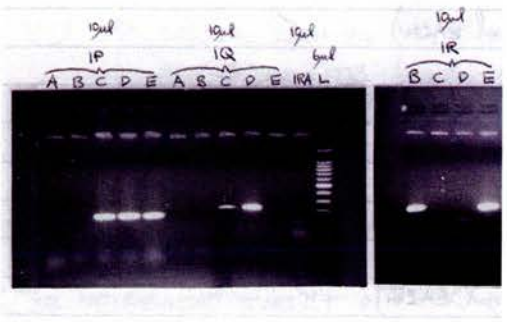


Figure B.2.1: Determining the presence and orientations of inserts in five colonies using PCR (*A* to *E*).

group	primers
1P	$long_L$ and $long_R$
1Q	$long_L$ and *SP6*
1R	$long_R$ and *SP6*

L = 100bp ladder (Promega);
IRA = group *1R*, colony *A*.

In group *1P*, the results for colonies *C*, *D* and *E* showed a clear amplification product of about 150bp verifying the presence of the "longmer". Clear bands were also visible for *C* and *D* in group *1Q* showing that the "longmer" was oriented towards the SP6 promoter. A clear band can be seen for *E* in group *1R* indicating that this insert was oriented in the opposite direction to *C* and *D*. Colony *B* had a successful result in group *1R*, but not in group *1P*; it is possible that either the left side of *B*'s "longmer" insert became mutated or that the corresponding group *1R* PCR reaction failed. A very faint band can also be seen for *A* in group *1P* but in neither of the other two groups.

Two colonies with inserts pointing in different directions were required for the second construction stage (Section 5.1.2): The colonies with the clearest results, *D* and *E*, were chosen for use in subsequent steps (as plasmids *pS2b* and *pS2c*).

B.2.2 Extraction

During the second wetware construction stage (Section 5.1.2) it was necessary to insert a *bfp* gene into a *pGEM-T* vector and to ensure that the gene was not downstream of the vector's T7 promoter, otherwise it would be continually expressed in *E. coli*. The beginning of the *bfp* gene contains an *Xba*I restriction site and a *Sac*II restriction site lies nearby the T7 promoter. There is also a *Sal*I restriction site present on the side of the vector's cloning site that is opposite the T7 promoter. Three colonies (*A* to *C*) were picked that had been transfected with an amplified *bfp* gene and DNA from each was subjected to two separate double-digests to determine the presence of the insert and its orientation: *Xba*I and *Sac*II (group *1*) to excise a gene oriented towards the T7 promoter, and *Xba*I and *Sal*I (group *2*) to excise a gene downstream of the T7 promoter (Figure B.2.2).

Figure B.2.2: Using digestion to determine insert presence and orientation, in three colonies (*A* to *C*).

group	enzymes
1	*Xba*I and *Sac*II
2	*Xba*I and *Sal*I

L = 100bp ladder (Promega).

DNA from colony *C* produced a clear band of about 1000bp in group *1* which was assumed to contain the *bfp* gene. Colony *C* DNA, digested in group *2* tests, produced a 200bp band which was assumed to be DNA from the vector's *Sal*I restriction site up to its cloning site, combined with the beginning of the *bfp* gene, up to its *Xba*I site. These results indicated that *C* contained the target plasmid (*pS2a*). Conversely, DNA from colonies *A* and *B* produced bands of about 800bp in the group *2* tests but no visible bands in group *1*. Therefore, this indicated that the inserts in *A* and *B* were not oriented in the desired direction. *Sal*I exhibited both low efficiency and star activity in the group *2* tests. This is likely to be as it was used in a general purpose, double-digestion buffer

(Fermentas Y^+/ Tango) rather than by performing digestion in two steps.

B.2.3 Assessing Enzyme Activity

In order to discover potential DNA-arranging conditions which were favourable for both digestion with *Bpu*10I and ligation with T4 DNA ligase, an exploratory experiment was performed to assess the relative effectiveness of *Bpu*10I over a range of conditions (Figure B.2.3). The DNA used was a *pGEM-T* vector containing a "longmer" (Section 5.1.1), with two *Bpu*10I restriction sites and an AFP insert.

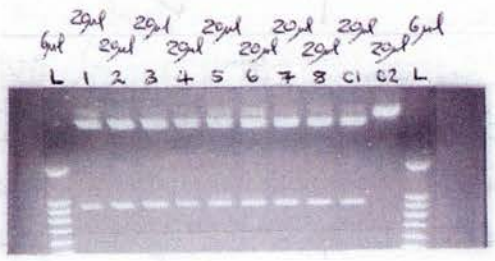


Figure B.2.3: Assessing the effectiveness of *Bpu*10I when incubated with PEG.

no.	buffer (Fermentas)	temp.
1	*Bpu*10I unique, 10% PEG	4°C
2	*Bpu*10I unique, 10% PEG	22°C
3	*Bpu*10I unique, 10% PEG	30°C
4	*Bpu*10I unique, 10% PEG	37°C
5	T4 DNA ligase, 10% PEG	4°C
6	T4 DNA ligase, 10% PEG	22°C
7	T4 DNA ligase, 10% PEG	30°C
8	T4 DNA ligase, 10% PEG	37°C
C1	*Bpu*10I unique	37°C
C2	Distilled water	37°C

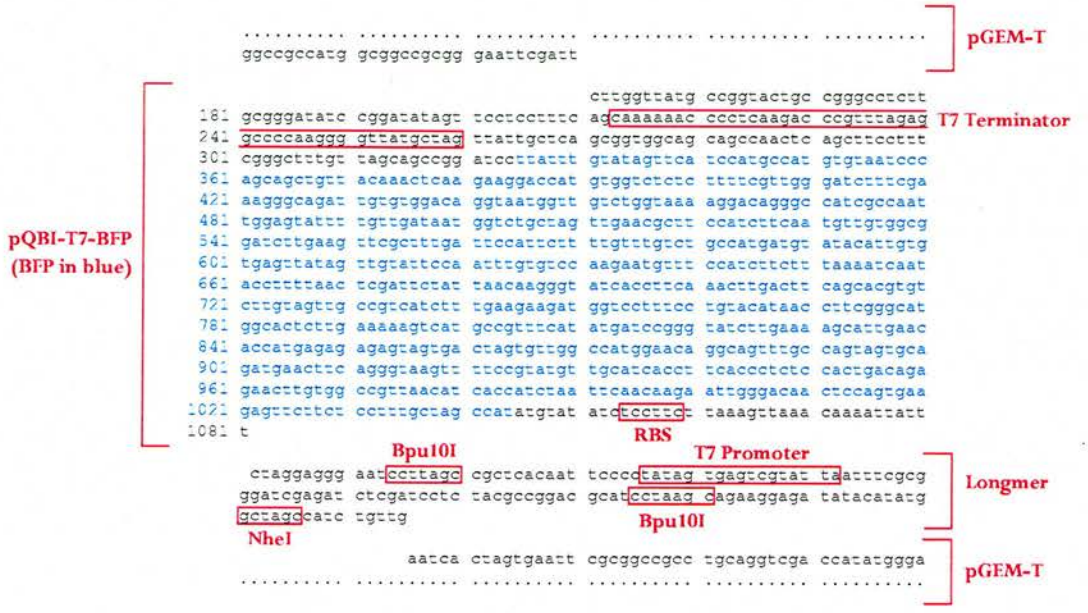
L = 100bp ladder (Promega).

The results were grouped into three relative categories: *complete digestion* (2,3 and 4); *almost complete digestion* (1,7,8,C1); and *partial digestion* (5 and 6).

B.3 Sequences

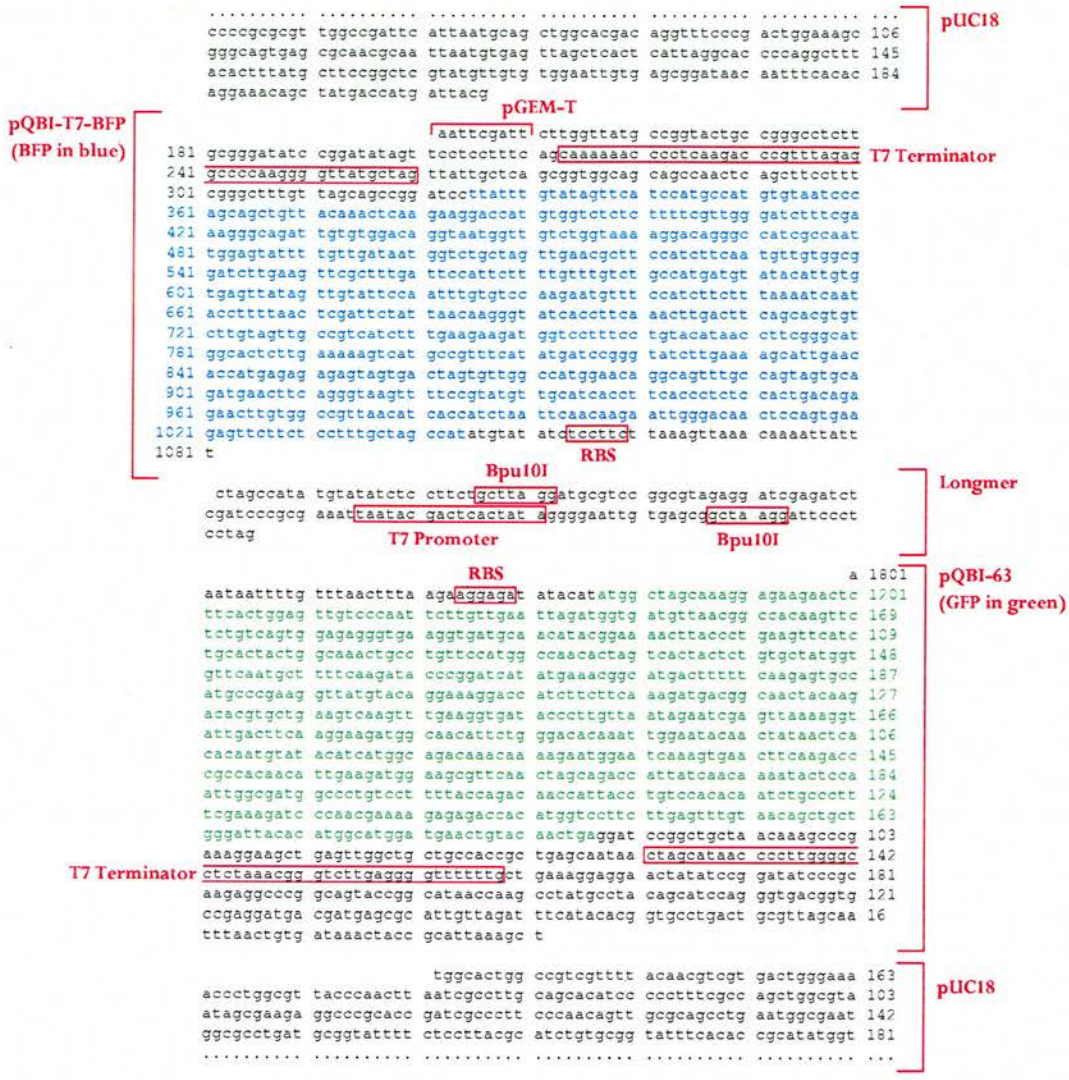
This section contains the nucleotide base sequences of target plasmids in Section 5.1.

B.3.1 Plasmid *pS2d*



B.3.2 Plasmid *pS2e*

B.3.3 Plasmid *pS3b*



Appendix C

Towards Programmable *In Vivo* Computation

D.K. Arvind and Marc Blenkiron
Institute for Computing Systems Architecture
School of Informatics, The University of Edinburgh
Mayfield Road, Edinburgh, EH9 3JZ, Scotland.
Email: dka|marcb@dcs.ed.ac.uk

June 1st 2003

Abstract

This paper is concerned with *in vivo* computation, i.e., the idea of using living cells, such as bacteria, as general computational devices, albeit slow ones. We speculate on ways in which functional units such as timers, counters, random number generators and programmable memory could be realised, and how one could engineer a class of cells that could then be programmed by external stimuli to trigger a desired computation. A notation has been devised for describing sequences of DNA in terms of variables representing units such as promoters, operators and messenger proteins and also provides a means to specify how a polymerase interacts with these sequences. A bespoke simulation environment, BioSim, will be used to evaluate and search the utility of emergent genetic networks, before testing them in wetware.

1 Introduction

Biological organisms continuously sense their environment and react to both internal and external stimuli, which can be exploited in the design of biological sensors. Weiss et al, for instance, have adopted the approach of engineering precise and reliable *in vivo* logic circuitry [WBKKMN02], which is embedded in the cells to process the information. In this paper we propose an alternative approach, which rather than imposing a strict digital regime, exploits the analogue and probabilistic behaviour of the cellular mechanisms to devise programmable *in vivo* computation fabrics.

The DNA of a living organism contains the totality of information required for its life-cycle, which is encoded in sequential blocks, called the *genes*. Whereas the DNA remains locked inside the nuclei within the chromosomes, some of its information can be imparted to a *messenger RNA (mRNA)* copy, from which proteins can be made. The DNA sequence is first *transcribed*, i.e., read, and then *translated*, i.e., converted into proteins. The genetic code may *express* or *repress*, i.e., turn on or off, respectively, particular proteins when required. The expression is controlled by a promoter which avoids the synthesis of unnecessary products. Control regions in the promoter, called *operators*, in the form of binding sites for regulator proteins, can up- or down-regulate transcription. The proteins produced from genes decay at different rates.

Enzymes called *polymerases* are responsible for the reading of DNA from a *promoter* to a *terminator* site. These enzymes can be modified when passing over a *utilisation* site, given the presence of a required protein. A modification might allow a *polymerase* to ignore a particular *termination* site and read beyond it.

We introduce the bacterial virus, phage λ, that is used to illustrate our examples. An attraction of studying such viruses is that their DNA sequence is typically shorter than that of other forms of life. This well-studied virus is able to infect a host bacterium, E. Coli, and can decide whether to combine itself with its host's DNA or make many copies of itself and break open its host, through a competition of *expression* between two genes, *cI* and *cro*.

When designing new genetic networks, as opposed to modifying existing ones, it can be difficult to find the correct set of proteins and sites (out of thousands that occur naturally) to produce the desired behaviour. A number of different options have to be tested in the target environment. We have to bear in mind that cells have evolved to suit their survival and reproduction, not to run our computational instructions. Cells have defence mechanisms that can spot certain sequences of DNA that should not be there, and destroy them. The cells could mutate to give them a selective advantage such as reducing the burden on its resources, if we do not guard against this.

In the rest of this paper, Section 2 presents a notation for describing genetic networks; we propose synthetic genetic networks that could form the basis for functional units, and speculate on ways to use them as analogue components and programming them. We briefly describe the BioSim simulation environment for modelling and simulating the behaviour of synthetic genetic networks for *in vivo* computation. Section 4 makes reference to wetware issues of concern, with concluding remarks in Section 5.

2 Notation

We present a notation for defining relationships between DNA, the action of polymerases, proteins, and transcripts. A statement in this notation can contain variables which can be assigned to sets of proteins or sites, from a database of experimental properties and ranges. The notation has arisen, in part, to aid in communicating with biologists, and has since been formalised to enable the modelling, analysis and prediction of biological processes. It is a particularly difficult task, in general, to choose suitable proteins and sites with the desired properties and interactions. We have available quite detailed information regarding some protein shapes, their reaction sites, and host environmental characteristics, and a rich database of experimental results to draw upon. Yet it is very unlikely that the instantiated components will interact in the desired manner, first time. Weiss et al [WBKKMN02] are developing a genetic tool box of optimised components which is a first step in this direction.

2.1 Format

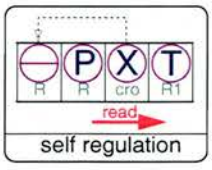
2.1.1 Γ, a tape

$\Gamma = [\ \mathbf{T}_{id} \mid \mathbf{P}_{id} \mid \oplus_{id} \mid \ominus_{id} \mid \mathbf{U}_{id} \mid \mathbf{X}_{id}\ ]^*$

Symbol	Meaning
$\mathbf{T}_{id}$	terminator
$\mathbf{P}_{id}$	promoter
$\oplus_{id}$	up-regulator (operator)
$\ominus_{id}$	down-regulator (operator)
$\mathbf{U}_{id}$	utilisation site
$\mathbf{X}_{id}$	gene (for protein transcription)

A tape, of type, Γ, is an ordered list of components made up of terminators, promoters, operators, utilisation sites and gene encodings. Each component is identified by a name, *id*. Any two components of the same type and with the same name are considered to have the same molecular composition. The name of a promoter will begin with either an L or R to indicate the direction of transcription. We draw on a small part of the genetic code of λ to illustrate a short tape in this notation.

$\Gamma_{sample} = \ominus_R\ \mathbf{P}_R\ \mathbf{X}_{cro}\ \mathbf{T}_{R1}$



Promoter, $\mathbf{P}_R$, and terminator, $\mathbf{T}_{R1}$, mark the start and the end of transcription, respectively. During transcription, a gene, $\mathbf{X}_{cro}$, produces an *mRNA* template from which the protein *cro* is made. This attaches itself to the operator site, $\ominus_R$, which down-regulates the promoter, $\mathbf{P}_R$. This process hinders further transcription. The products of *mRNA* and *cro* are subject to decay and thus the negative feedback loop regulates the level of *cro*.

2.1.2 Φ, a polymerase

$\Phi = [(\mathbf{P}_{id}, Q)]^*, [(\ominus_{id}, M_{id}, \mathbf{P}_{id}, Q)]^*, [(\oplus_{id}, M_{id}, \mathbf{P}_{id}, Q)]^*, [(\mathbf{T}_{id}, Q)]^*, [(\mathbf{U}_{id}, \mathbf{T}_{id}, Q)]^*, \kappa_\Phi$

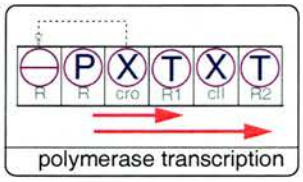
Symbol	Meaning
M_{id}	A type of protein
M_{null}	The absence of a protein
Q	$x \in Q \iff 0 \leq x \leq 1$
κ_Φ	The cost of making this enzyme

Group	Meaning
$(\mathbf{P}_{id}, Q)$	Affinities for promoters
$(\ominus_{id}, M_{id}, \mathbf{P}_{id}, Q)$	Up regulation effect
$(\ominus_{id}, M_{id}, \mathbf{P}_{id}, Q)$	Down regulation effect
$(\mathbf{T}_{id}, Q)$	Terminator effectiveness
$(\mathbf{U}_{id}, \mathbf{T}_{id}, Q)$	Utilisation site's effect

A polymerase of type, Φ, describes the behaviour of a transcribing enzyme, such as the chances of it starting transcription from a particular place, and how operators and terminators may interfere. Given a protein is attached to a utilisation site and a polymerase passes over it during transcription, we can encode a change in the polymerase, such as the reduction in the effectiveness of a termination site. The following example illustrates part of the behaviour of the E. Coli polymerase, as it interacts with a simplified short

tape corresponding to a longer sequence of phage λ.

Φ_{sample} = (P_R, 0.95), ($\ominus_R$, M_{cI}, P_R, 1.0), ($\ominus_R$, M_{cro}, P_R, 0.7), (T_{R1}, 0.5), (T_{R2}, 1.0)



This polymerase has a strong affinity for P_R and therefore has a high chance of starting transcription from here. But when a M_{cro} is in $\ominus_R$, this promoter is partially blocked, which lowers its chance. The terminator, T_{R1}, is only effective half of the time, whereas the terminator, T_{R2}, is effective all of the time. In other words, half of the time, X_{cro} will produce a transcript, and in the other half, a transcript from both X_{cro} and X_{cII} will be produced.

2.1.3 M, a protein type

M = [(P_{id}, Q)]*, [($\ominus_{id}$, Q)]*, [($\ominus_{id}$, Q)]*, [(U_{id}, Q)]*, δ_{att}, δ_{un}, κ_M

Symbol	Meaning
δ_{att}	The rate of decay when *attached*
δ_{un}	The rate of decay when *unattached*
κ_M	The cost of making this protein

Group	Meaning
(P_{id}, Q)	Affinity for promoter activation site
($\ominus_{id}$, Q)	Affinity for up-regulating operator
($\ominus_{id}$, Q)	Affinity for down-regulating operator
(U_{id}, Q)	Affinity for utilisation site

This statement describes a protein of type, M, with its affinities for promoters, operators and utilisation sites, decay rates and cost of its production. Catalysts and other effects on affinities are not directly stated.

2.1.4 X, a transcript encoding type

X = [(M_{id}, τ)]*, δ

Symbol	Meaning
τ	A length of time
δ	Decay rate of its *mRNA*

Group	Meaning
(M_{id}, τ)	A protein type and its length

This statement lists a transcription encoding of type, X, which lists a number of protein products and their duration in time units, which gives an indication of how long the processes of transcription and translation take. A decay rate provides an indication of the stability of the *mRNA* template that is produced by transcription. For example:

X_{sample} = (M_{cro}, 20), (M_{cII}, 10), 0.05

So the transcript for M_{cII} would take 10 time units and the whole process would take 30.

2.1.5 Relationships

A set of relationships which describes the low-level interactions in a system may be specified by:

$[T_{id}]^*$, $[\Phi_{id}]^*$, $[M_{id}]^*$, $[X_{id}]^*$

This statement describes a collection of tape configurations, and types of polymerase, proteins, and transcription. Note that a particular instance of a polymerase may be modified when it passes over a utilisation site, and that an instance of a tape configuration may be modified, for example, by mutation.

2.2 Cellular Functional Units

We next speculate on how functional units, such as counters, memory and random number generators, could be implemented in cellular fabric in order to realise programmable *in vivo* computation. Other functional units, such as ring oscillators, have already been demonstrated [ElLe00]. It is very likely that once we begin to test even the most simple of these building blocks in wetware, then we would need to significantly tune and adjust the design and parameters in order to achieve their desired operation. A number of simplifications have been made to aid the clarity of presentation: in particular, it is often desirable that a protein is able to negatively regulate its own production, which would lead to a more stable level of concentration and not exhaust the host cell's resources.

2.2.1 Random Number Generator

A random number is considered to be a sequence of booleans, each assigned either true or false. The

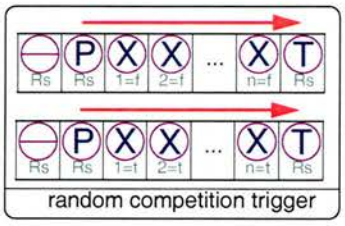
method postulated here would undoubtedly be subject to environmental biases. Let us consider a number with m booleans of form:

$$\sum_{i=0}^{m-1} 2^i \nu_i$$

Each ν_i will have a specific protein encoded to true, $\mathbf{X}_{i=T}$, and a specific protein encoded to false $\mathbf{X}_{i=F}$. At a trigger point, the following sequences will be transcribed for a short burst of time:

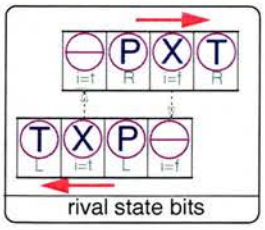
$\ominus_{Rs}\mathbf{P}_{Rs}\mathbf{X}_{0=T}\mathbf{X}_{1=T}\ \ldots\ \mathbf{X}_{m-1=T}\mathbf{T}_{Rs}$

$\ominus_{Rs}\mathbf{P}_{Rs}\mathbf{X}_{0=F}\mathbf{X}_{1=F}\ \ldots\ \mathbf{X}_{m-1=F}\mathbf{T}_{Rs}$



which would cause approximately similar quantities of true and false values for each variable to be present. A protein will be able to inhibit the production of its counterpart value, leading to a competition, such that after a period of time, the number will stabilise. The plausibility of such a two-state component has already been demonstrated in wetware [GCC00].

$\ominus_{i=T}\mathbf{P}_R\mathbf{X}_{i=F}\mathbf{T}_R\mathbf{T}_L\mathbf{X}_{i=T}\mathbf{P}_L\ominus_{i=F}$



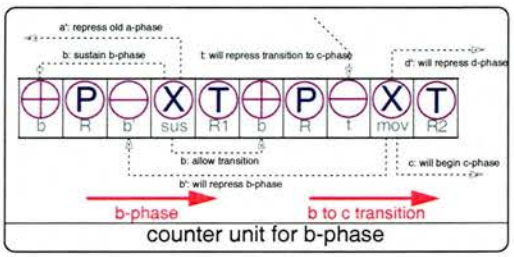
2.2.2 Counters

We may wish to count the number of times an event occurs, such as cell division, or perform a function every n generations. Our event trigger is a form of signal that is generated at intervals as a protein that also degrades rapidly.

In our case, a counter goes through a series of possible phases, where a diffuse signal fires an increment to the next phase. Although subject to behavioural fluctuations, we would wish to ensure that the counter does not increment by more than a phase for a given trigger, and that only one phase is ever dominant for a length of time. Consider:

$\pi(a, b, c, d, \Omega) =$
$\ominus_b\mathbf{P}_R\ominus_{b'}\mathbf{X}_b\mathbf{X}_{a'}\mathbf{T}_{R1}\,\Omega_t\ \ominus_b\mathbf{P}_R\mathbf{X}_{d'}\mathbf{X}_{b'}\mathbf{X}_c\mathbf{T}_{R2}$

where a, b, c, and d are the four phases, and b is the current phase. Ω is one of the two types of operators.



When in phase b, production of b is maintained. The previous phase, a is suppressed by production of a'. When an event signal, corresponding to protein M_l is either bound to the Ω site or is absent through decay, then three different proteins are produced: b' suppresses the production of b, c starts the move to the next phase, and d' ensures that the phase does not change twice on the same trigger by inhibiting the phase following c.

A four phase counter could be represented by:

$\pi(a, b, c, d, +)\ \pi(b, c, d, a, -)$
$\pi(c, d, a, b, +)\ \pi(d, a, b, c, -)$

2.2.3 Timers

Given that a class of cells already has a counter and an oscillator to drive it, it would then be possible to approximate the synchronisation of a number of cells' counters. This can be achieved by passing particles between cells, as has been demonstrated in [McKoHaCo02]. On cell division, the phase of the clock would be passed on to the two resulting cells through the protein concentrations from the original cell.

2.2.4 Memory

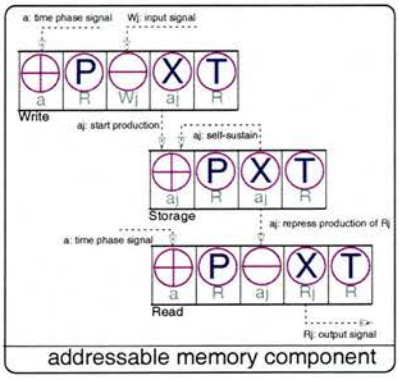
We next propose the design of a simple serial write, serial read cyclic tape storage.

- The tape consists of a cycle n squares, where each square, $\sigma_{0 \leq i < n}$, corresponds to a point in an n phase counter. Each square can hold m locations.

- There are m distinct *write* location signals, each being encoded in a protein $\mathrm{M}_{Wj:(j<m)}$. Each signal corresponds to a location offset. There are m distinct *read* location signals, each encoded in a protein $\mathrm{M}_{Rj:(j<m)}$. Each signal corresponds to a location offset.

- At counter time, $i < n$, the protein, ρ_i, will be present inside the cell as produced by the timer. The presence of a write signal, M_{Wj}, will cause the square σ_i to have its j^{th} location to be set. Also at counter time, $i < n$, if σ_i has its j^{th} location set, then the read signal M_{Rj} will be produced.

Here is a partial sketch of the implementation. We acknowledge that there may be a degree of over-spill of reads and writes tied to a timer:



- Write, $\oplus_{\alpha}\mathbf{P}_{R}\ominus_{Wj}\mathbf{X}_{\alpha_j}\mathbf{T}_{R}$, when time signal α is active and there is no presence of the write signal, M_{Wj}, then some M_{α_j} will be produced.

- Storage, $\oplus_{\alpha_j}\mathbf{P}_{R}\mathbf{X}_{\alpha_j}\mathbf{T}_{R}$, the signal protein, M_{α_j}, is self-sustaining. Regulation of the level of this protein is glossed over.

- Read, $\oplus_{\alpha}\mathbf{P}_{R}\ominus_{\alpha_j}\mathbf{X}_{Rj}\mathbf{T}_{R}$, at time, α, the read protein, M_{Rj}, is produced only if there is no α_j bound to the down operator.

One could imagine a partially associative form of such a memory where a particular write signal activates a *number* of memory locations with *differing probabilities*. In this way we may also be able to create multiple location instructions.

2.2.5 Digital or Analogue?

Signals in cellular networks take the form of protein concentrations. We do not necessarily need to utilise these as digital-type components. For example, a number of processes may contribute to the production of proteins, which depending probabilistically on their *relative* levels will cause an action, or a more graded analogue response, such as movement. In the phage λ virus that infects E. Coli bacteria, for example, a decision is made whether or not to lay dormant in the host's DNA and multiply with it, or to take over the cell and multiply rapidly, thereby killing the host through a competition over promoter and operator sites by the relative levels of two signals, *cI* and *cro*. In addition, there also exist sets of proteins and sites that have differing affinities for each other.

2.3 A Case Study in Programmability

We present an example which illustrates *in vivo* computation to solve a boolean variable satisfaction problem. We describe how a sequence of DNA could be added, perhaps by virus, to a cell and cause the *utilisation* of a previously engineered functional units.

Given a CNF statement, ψ, with n clauses:

$$\psi = \bigwedge_{i=0}^{n-1} \omega_i$$

where, for ease of illustration, each clause contains two out of m possible boolean variables in one of the following combinations:

$$\omega_i = (\nu_j \vee \nu_k) \quad and \quad (\overline{\nu}_j \vee \nu_k)$$

$$and \quad (\nu_j \vee \overline{\nu}_k) \quad and \quad (\overline{\nu}_j \vee \overline{\nu}_k) \quad : \quad i < n; j, k < m$$

We wish to discover a sequence of boolean assignments to each of the m variables, such that ψ is true. We may negate both sides of the equation to obtain:

$$\overline{\psi} = \bigvee_{i=0}^{n-1} \overline{\omega}_i$$

Therefore, if any one of the negated clauses is found to be true, then the whole expression will be false. Clauses are converted to the following:

$$\overline{\omega}_i = (\overline{\nu}_j \wedge \overline{\nu}_k) \quad or \quad (\nu_j \wedge \overline{\nu}_k)$$

$$or \quad (\overline{\nu}_j \wedge \nu_k) \quad or \quad (\nu_j \wedge \nu_k) \quad : \quad i < n; j, k < m$$

2.3.1 Translation

Each clause may be mapped to the notation as follows:

$$\begin{array}{ll} \overline{\nu}_j \wedge \overline{\nu}_k & \ominus_k \mathbf{P}_R \ominus_j \mathbf{U}_z \mathbf{T}_z \mathbf{X}_p \mathbf{T}_e \\ \nu_j \wedge \overline{\nu}_k & \ominus_k \mathbf{P}_R \oplus_j \mathbf{U}_z \mathbf{T}_z \mathbf{X}_p \mathbf{T}_e \\ \overline{\nu}_j \wedge \nu_k & \ominus_j \mathbf{P}_R \oplus_k \mathbf{U}_z \mathbf{T}_z \mathbf{X}_p \mathbf{T}_e \\ \nu_j \wedge \nu_k & \oplus_j \mathbf{P}_R \oplus_k \mathbf{U}_z \mathbf{T}_z \mathbf{X}_p \mathbf{T}_e \end{array}$$

where, $\mathbf{T}_z$ encodes a terminator that is skipped if utilisation site, $\mathbf{U}_z$, has a protein attached. Terminator, $\mathbf{T}_e$, is a termination site even for a modified polymerase. Each $\ominus_{j,k}$ encodes an operator that blocks its adjacent promoter if a corresponding j, k protein is attached. Each $\oplus_{j,k}$ encodes an up-regulator that requires activation by a corresponding j, k protein. We note that there are relatively few up-regulators that work well in conjunction, however, this can be resolved by causing the presence of a protein, j to block the production of an intermediate protein, j' thus allowing the use of j' and k in conjunction. $\mathbf{X}_p$ could encode a protein that kills the cell or, less dramatically, resets the counter. A correct guess, noted by not having generated protein $\mathbf{X}_p$ after another counter run, could cause the production of a fast-decaying protein that renders it immune to a virus that could be used to infect a lawn of bacteria to select the candidate answers.

2.3.2 Execution

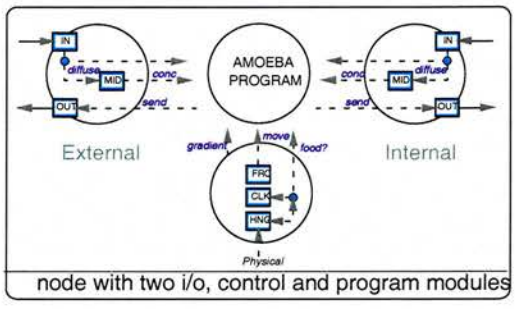
Using a random number generator as described in Section 2.2.1 of size m locations for each variable, a guess is made for a correct set of assignments. While this happens, a counter will tick for a period of time which is sufficient for the random number generator to stabilise. Once the counter reaches its end value, then proteins will be transcribed that bind to the utilisation sites $\mathbf{U}_z$. Any incorrect assignments could result in dead, or reset bacteria. The remaining bacteria will either have guessed the correct assignment, or some other factor, such as mutation, may have interfered.

3 Simulation Environment

3.1 Platform

BiSim is a simulation platform which models the biological cell as a network of functional units at different levels of abstraction. Provisions have been made to efficiently support complex, time-changing models and the BiSim engine's (soft virtual machine, event-queue based) design is tailored to run on a network of workstations such as a Beowulf cluster, as is its companion directed-search tool that is used in optimisation.

The support for modularity and flexibility allows devices to have multiple, time- and event-varying channels for inter-cellular and intra-cellular communication, with the ability to express characteristics such as external effects on signal strength or chemical diffusion characteristics and supports their use in a script file. For example, we may choose to model particle reactions using concentrations and rates, as stochastic kinetic reactions, or a combination through specification in an interpreted script file. Notably, script files allow *ranges* of settings, to provide for batch runs and directed search over numerical and structural parameters.



node with two i/o, control and program modules

Nodes in such a network can range from collections of cells to cellular pathway junctions, which allows a simulation to take place in a mixture of different levels of abstraction for convenience and efficiency. A component of a node specifies its input/output, storage and environmental interactions. A number of exported instructions may be declared which allows a node's complete behaviour to be expressed in these terms.

4 Wetware

4.1 "Hardware"

We propose that sequences representing functional units will be encoded inside a viral plasmid, such as T4, which will be inserted into our bacterial cell, such as the well-studied E. Coli. We propose that these ready-made computational devices could be stored and later used *off the shelf* for a desired computation, for example, by storing them in liquid nitrogen, an established practice. A slow and incremental approach will be necessary in creating functional units of even slight complexity.

4.2 Corruption

One of the many problems facing a hardware or software encoding in DNA will be corruption, whether from UV light or another source of mutation. If small proportions of cells were, in this respect, to malfunction, this is relatively straight forward to tackle, at least in principle, using means such as redundancy in groups. If however, a bacterium happened to mutate such that it reduced the length of its DNA, causing it a selective advantage, but also corrupting a functional unit, and then multiplying, this would cause a different type of problem. We plan to investigate whether analogues of error-checking as used frequently in computer communications could be applied to DNA messages - perhaps causing cell death if discovered. Graph colouring algorithms have already been illustrated as a means to add robustness [AmGiHo96] to a DNA based computer [Adelman96] and bacteria and viruses themselves can produce enzymes that recognise certain unwanted, short sequences of DNA and destroy them.

4.3 "Software"

Our colony of loosely synchronised bacterial cells may emit some perceivable signal each time their clock cycles wrap, such as a visible dye. We can send in modified sugars that can cause activation of genes inside cells to effectively generate external signals. We can encode a form of corruption checking by causing the removal of certain code to cause the production of an antibiotic such as ampicillin that will kill the bacterium. Similarly, we can send in our software encoded in a stretch of tamed λ or another virus that will not adversely interact with the previous method of inserting hardware functional units. We are investigating the possibilities of removal or deactivation of one software program for another, such as by selectively destroying the existing program's DNA and preventing the host bacterium from intervening (disabling the SOS response) for a period of time to provide a form of reprogrammability.

5 Conclusions

We have presented an alternative approach to programmable *in vivo* computation which exploits the analogue and probabilistic nature of cellular processes. A notation has been devised for describing genetic networks, with designs presented for functional units such as counters, memory and random number generators. The B#Sim simulation environment has been implemented and tested. Future work will investigate the implementation of the functional unit designs in wetware and refining the simulation models to be able to accurately predict their behaviour in the B#Sim environment.

Acknowledgements

We would like to thank Jamie Davies, Department of Biomedical Sciences, University of Edinburgh for discussions on the wetware aspects of *in vivo* computation.

References

[Adelman96] Adleman, L. 1996. *On constructing a molecular computer*, DNA Based Computers, Eds. R. Lipton and E. Baum. DIMACS: series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society. 1-21(1996).

[AmGiHo96] Amos, M., Gibbons, A., and Hodgson, D. 1996. *Error-resistant implementation of DNA Computations*, Proceedings of the Second Annual Meeting on DNA Based Computers, Princeton, USA.

[McKoHaCo02] McMillen, D., Kopel N., Hasty, J., and Collins, J. 2002. *Synchronizing genetic relaxation oscillators by intercell signaling*, Proceedings of the National Academy of Science 99, 670-684.

[ElLe00] Elowitz, M. B. and Leibler, S. 2000. *A synthetic oscillatory network of transcriptional regulators*, Nature 403, 335-338.

[WBKKMN02] Weiss, R., Basu, S., Kalmbach, A., Karig, D., Mehreja, R., and Netatvali, I. 2002. *Genetic Circuit Building Blocks for Cellular Computation, Communications and Signal Processing*, International Journal of Natural Computing, Kluwer, In Press.

[GCC00] Gardener, T., Cantor, C., and Collins, J. 2000. *Construction of a genetic toggle switch in Escherichia coli*, Nature 403, 339-342.

Bibliography

[Adleman, 1996] Adleman, L. *On constructing a molecular computer*, DNA Based Computers, Eds. R. Lipton and E. Baum. DIMACS: series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society. 1-21.

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

[Amos, 2004] Amos, M. (Ed.) *Cellular Computing.* Series in Systems Biology, Oxford University Press, USA.

[Amos *et al.*, 2006] Amos, M., Hodgson, D.A., Gibbons, A. *Bacterial self-organisation and computation.* International Journal of Unconventional Computing, in press.

[Arkin and Ross, 1994] Arkin, A., Ross, J. *Computational functions in biochemical reaction networks.* Biophys J. Aug;67(2):560-78.

[Arvind and Blenkiron, 2003] Arvind, D. K., Blenkiron, M. *Towards Programmable In Vivo Computation*, Proceedings of the Second Annual Workshop on Non-Silicon Computation, San Diego, USA. Available on-line at `http://blenkiron.com/phd/`.

[Babbage, 1864] Babbage, C. *Passages from the Life of a Philosopher: Of the Analytical Engine, Chapter VIII.*

[Bassler, 1999] Bassler, B.L. *How bacteria talk to each other: regulation of gene expression by quorum sensing.* Curr Opin Microbiol. Dec;2(6):582-7.

[Basu *et al.*, 2004] Basu, S., Mehreja, R., Thiberge, S., Chen, M., Weiss, R.. *Spatiotemporal control of gene expression with pulse-generating networks* PNAS. April 27. Vol. 101, No. 17, 6355-6360.

[Basu *et al.*, 2005] Basu, S., Gerchman, Y., Collins, C.H., Arnold, F.H., Weiss, R. *A synthetic multicellular system for programmed pattern formation.* Nature. Apr 28;434(7037):1130-4.

[Batt *et al.*, 2005] Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., Schneider, D. *Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in Escherichia coli.* Bioinformatics. Jun 1;21 Suppl 1:i19-i28.

[Becskei and Serrano, 2000] Becskei, A., Serrano, L. *Engineering stability in gene networks by autoregulation.* Nature. Jun 1;405

[Belfort and Roberts, 1997] Belfort, M., Roberts, R.J. *Homing endonucleases: keeping the house in order.* Nucleic Acids Res. Sep 1;25(17):3379-88.

[Benenson *et al.*, 2003] Benenson Y., Adar R., Paz-Elizur T., Livneh Z., Shapiro E. *DNA molecule provides a computing machine with both data and fuel.* Proc Natl Acad Sci U S A. Mar 4;100(5):2191-6.

[Benenson *et al.*, 2004] Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E. *An autonomous molecular computer for logical control of gene expression.* Nature. 27;429(6990):423-9.

[Benner and Sismour, 2005] Benner, S. A. and Sismour, M *Synthetic Biology.* Nat Rev Genet. Jul;6(7):533-43.

[Berg, 2004] Berg, H.C. *E. coli in Motion.* Springer, Biological and Medical Physics Series.

[Bird, 1992] Bird, A. *The essentials of DNA methylation.* Cell 70, 5-8.

['BioBricks'] *MIT Registry of Standard Biological Parts*, `http://parts.mit.edu/`. Accessed December 2005.

[Blake *et al.*, 2003] Blake, W., Kærn, M., Cantor, C., Collins, J. *Noise in eukaryotic gene expression.* Nature. Apr 10;422(6932):633-7.

[Blakemore, 1975] Blakemore, R. *Magnetotactic bacteria.* Science, 190, 377-379.

[Blattner *et al.*, 1997] Blattner, F.R., Plunkett, G., Bloch, C.A., Perna, N.T., Burland, V., Riley, M., Collado-Vides, J., Glasner, J.D., Rode, C.K., Mayhew, G.F., Gregor, J., Davis, N.W., Kirkpatrick, H.A., Goeden, M.A., Rose, D.J., Mau, B., Shao, Y. *The complete genome sequence of Escherichia coli K-12.* Science. Sep 5;277(5331):1453-74.

[Blenkiron, 2004] Blenkiron, B., *BioSim User Guide.* Available on-line at `http://blenkiron.com/phd/`.

[Bray, 1995] Bray, D. *Protein molecules as computational elements in living cells.* Nature. Jul 27;376(6538):307-12

[Bray and Lay, 1994] Bray, D., Lay, S. *Computer simulated evolution of a network of cell-signaling molecules.* Biophys J. Apr;66(4):972-7

[Burt, 2003] Burt, A. *Site-specific selfish genes as tools for the control and genetic engineering of natural populations.* Proc Biol Sci. May 7;270(1518):921-8.

[Burt and Koufopanou, 2004] Burt, A., Koufopanou, V. *Homing endonuclease genes: the rise and fall and rise again of a selfish element.* Curr Opin Genet Dev. Dec;14(6):609-15. Review.

[Campbell-Kelly, 2004] Campbell-Kelly, M. *Computer: A History of the Information Machine.* Perseus, Second Edition.

[Chevalier *et al.*, 2002] Chevalier, B.S., Kortemme, T., Chadsey, M.S., Baker, D., Monnat, R.J., Stoddard, B.L. *Design, activity, and structure of a highly specific artificial endonuclease.* Mol Cell. Oct;10(4):895-905.

[Clarke *et al.*, 1999] Clarke, E., Grumberg, O. and Peled, D. *Model Checking.* MIT Press, Cambridge, MA.

[Collins *et al.*, 2003] Collins, C.H., Yokobayashi, Y., Umeno, D., Arnold, F.H. *Engineering proteins that bind, move, make and break DNA.* Curr Opin Biotechnol. Aug;14(4):371-8.

[Cook, 1971] Cook, S.A. *The complexity of theorem proving procedures* Proceedings, Third Annual ACM Symposium on the Theory of Computing, ACM, New York, 151-158.

[Crick, 1958] Crick, F.H. *On protein synthesis"* Symp Soc Exp Biol. ;12:138-63.

[Dueber *et al.*, 2003] Dueber, E. J., Yeh, B. J., Chak, K., and Lim, W. A. *Reprogramming Control of an Allosteric Signaling Switch Through Modular Recombination* Science, Vol 301, Issue 5641, 1904-1908, 26 September

[Ehrenfeucht *et al.*, 2004] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D.M., Rozenberg, G. *Computation in Living Cells: Gene Assembly in Ciliates* Springer Natural Computing Series, XIV.

[Elowitz and Leibler, 2000] Elowitz, M. B. and Leibler, S. *A synthetic oscillatory network of transcriptional regulators*, Nature 403, 335-338.

[Elowitz *et al.*, 2002] Elowitz, M., Levine, A., Siggia, E., Swain, P. *Stochastic gene expression in a single cell.* Science. Aug 16;297(5584):1183-6.

[Endy and Brent, 2001] Endy, D. and Brent, R. *Modeling cellular behaviour*, Nature 409, 391-395.

[Endy, 2005] Endy, D. *Foundations for engineering biology.* Nature. Nov 24;438(7067) 449-53.

[Eriksson, 2002] Eriksson, S., Hurme, R., Rhen, M. *Low-temperature sensors in bacteria.* Philos Trans R Soc Lond B Biol Sci. Jul 29;357(1423):887-93.

[Fleischmann *et al.*, 1995] Fleischmann, R.D., Adams, M.D., White, O., Clayton, R.A., Kirkness, E.F., Kerlavage, A.R., Bult, C.J., Tomb, J.F., Dougherty, B.A., Merrick, J.M. *Whole-genome random sequencing and assembly of Haemophilus influenzae.* Science. Jul 28;269(5223):496-512.

[François and Hakim, 2004] François, P., Hakim, V. *Design of genetic networks with specified functions by evolution in silico.* Proc Natl Acad Sci U S A. Jan 13;101(2):580-5.

[Foster, 2001] Foster, J. *Evolutionary computation.* Nat Rev Genet. Jun;2(6):428-36.

[Fuqua *et al.*, 1994] Fuqua, W.C., Winans, S.C., Greenberg, E.P. *Quorum sensing in bacteria: the LuxR-LuxI family of cell density-responsive transcriptional regulators.* J Bacteriol. 1994 Jan;176(2):269-75.

[Gambardella *et al.*, 2003] Gambardella, P., Rusponi, S., Veronese, M., Dhesi, S.S., Grazioli, C., Dallmeyer, A., Cabria, I., Zeller, R., Dederichs, P.H., Kern, K., Carbone, C., Brune, H. *Giant Magnetic Anisotropy of Single Cobalt Atoms and Nanoparticles.* Science, May, Vol. 300, No. 5622, 1130-1133.

[Gardener *et al.*, 2000] Gardener, T., Cantor, C., and Collins, J. *Construction of a genetic toggle switch in Escherichia coli*, Nature 403, 339-342.

[Gibbs, 2004] Gibbs, W. *Synthetic Life.* Sci Am. May;290(5):74-81.

[Gillespie, 1977] Gillespie, D. T. *Exact stochastic simulation of coupled chemical reactions.* J. Phys. Chem. 81, 2340-2361.

[Gong and Golic, 2003] Gong, W.J., Golic, K.G. *Ends-out, or replacement, gene targeting in Drosophila.* Proc Natl Acad Sci U S A. Mar 4;100(5):2556-61.

[Gould and Subramani, 1988] Gould, S.J. and Subramani, S. *Firefly luciferase as a tool in molecular and cell biology.* Anal. Biochem. 175, 5?13.

[Guet *et al.*, 2002] Guet, C. C. , Elowitz, M. B. , Hsing, W., Leibler, S. *Combinatorial synthesis of genetic networks.* Science. May 24;296(5572):1466-70.

[Hasty *et al.*, 2002] Hasty, J., McMillen D., Collins, J. *Engineered gene circuits*, Nature 420, 224-230.

[Herskowitz and Hagen, 1980] Herskowitz, I. and Hagen, D. *The Lysis-Lysogeny Decision of Phage lambda: Explicit Programming and Responsiveness* Annual Review of Genetics, December , Vol. 14, Pages 399-445.

[Herskowitz, 1973] Herskowitz, I. *Control of Gene Expression in Bacteriophage Lambda.* Annual Review of Genetics, December , Vol. 7: Pages 289-324.

[Jackson *et al.*, 1972] Jackson, D.A., Symons, R.H., Berg, P. *Biochemical method for inserting new genetic information into DNA of Simian Virus 40: circular SV40 DNA molecules containing lambda phage genes and the galactose operon of Escherichia coli.* Proc Natl Acad Sci USA. Oct;69(10):2904-9.

[Jacob and Monod, 1961] Jacob, F., and Monod, J. *On the regulation of gene activity.* Cold Spring Harbor Symposia on Quantitative Biology 26, Cold Spring Harbor Laboratory, New York.

[Janeway, 2001] Janeway, C.A. *How the immune system works to protect the host from infection: a personal view.* Proc Natl Acad Sci USA. Jun 19;98(13):7461-8.

[Jasin, 1996] Jasin, M. *Genetic manipulation of genomes with rare-cutting endonucleases.* Trends Genet. Jun;12(6):224-8.

[Jong, 2002] de Jong, H. *Modeling and simulation of genetic regulatory systems: a literature review.* J Comput Biol. 9(1):67-103.

[Kærn *et al.*, 2004] Kærn, M., Blake, W.J., Collins, J.J. *The engineering of gene regulatory networks.* Annual Review of Biomedical Engineering, Vol. 5, 179-206.

[Kari and Landweber, 1999] Kari, L., Landweber, L.F. *Computational power of gene rearrangement.* In Proceedings of DNA Bases Computers, Winfree, V.E. and Gifford, D.K. (eds.), American Mathematical Society, 207-216.

[Kauffman, 1974] Kauffman, S. *The large scale structure and dynamics of gene control circuits: an ensemble approach.* J Theor Biol. Mar;44(1):167-90.

[Kelly and Smith, 1970] Kelly, T.J., Smith, H.O. *A restriction enzyme from Hemophilus influenzae. II.* J. Mol. Biol. 51, 393-409.

[Keseler *et al.*, 2005] Keseler, I., Collado-Vides, J., Gama-Castro, S., Ingraham, J., Paley, S., Paulsen, I., Peralta-Gil, M., Karp, D. *EcoCyc: a comprehensive database resource for Escherichia coli.* Nucleic Acids Res. Jan 1;33

[Kilby *et al.*, 1993] Kilby, N.J., Snaith, M.R., Murray, J.A. *Site-specific recombinases: tools for genome engineering.* Trends in Genetics, 9(12):413-21.

[Knight, 2001] Knight, T. *Idempotent Vector Design for Standard Assembly of Biobricks.* Unpublished, MIT.

[Kobayashi *et al.*, 2004] Kobayashi, H., Kaern, M., Araki, M., Chung, K., Gardner, T., Cantor, C. and Collins, J. *Programmable cells: Interfacing natural and engineered gene networks.* Proc. Natl. Acad. Sci. USA, 101(22):8414-8419

[Koch *et al.*, 2005] Koch, I., Junker, B., Heiner, M. *Application of Petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber.* Bioinformatics. Apr 1;21(7):1219-26.

[Koltsov and Perry, 2004] Koltsov, D., Perry, M. *Magnets and nanometres: mutual attraction.* Physics World, July, 17(7):31-37.

[Kramer, 2004] Kramer, B., Viretta, A., Baba, M., Aubel, D., Weber, W. and Fussenegger, M. *An engineered epigenetic transgene switch in mammalian cells.* Nature Biotechnology, 22:867-870.

[Kramers, 1940] Kramers, H.A. *Brownian motion in a field of force and the diffusion model of chemical reactions.* Physica 7: 284304.

[Kunst *et al.*, 1997] Kunst, F., Ogasawara, N., Moszer, I., Albertini, A.M., Alloni, G., Azevedo, V., Bertero, M.G., Bessieres, P., Bolotin, A., Borchert, S., Borriss, R., Boursier, L., Brans, A., Braun, M., Brignell, S.C., Bron, S., Brouillet, S., Bruschi, C.V., Caldwell, B., Capuano, V., Carter, N.M., Choi, S.K., Codani, J.J., Connerton, I.F., Danchin, A. *The complete genome sequence of the gram-positive bacterium Bacillus subtilis.* Nature. Nov 20;390(6657):237-8.

[Kuthan, 2001] Kuthan, H. *Self-organisation and orderly processes by individual protein complexes in the bacterial cell.* Prog Biophys Mol Biol. 75(1-2):1-17.

[Lakso *et al.*, 1992] Lakso M., Sauer B., Mosinger B. Jr, Lee E.J., Manning R.W., Yu S.H., Mulder K.L., Westphal H. *Targeted oncogene activation by site-specific recombination in transgenic mice.* Proc Natl Acad Sci U S A. Jul 15;89(14):6232-6.

[Le Novère and Shimizu, 2001] Le Novère, N., Shimizu, T.S. *STOCHSIM: modelling of stochastic biomolecular processes.* Bioinformatics. Jun;17(6):575-6.

[Lipkow *et al.*, 2005] Lipkow, K., Andrews, S.S. and Bray, D. *Simulated diffusion of phosphorylated CheY through the cytoplasm of Escherichia coli.* J Bacteriol. 2005 Jan;187(1):45-53.

[Lehmer, 1949] Lehmer, D.H. *Mathematical methods in large-scale computing units.* Proceedings of the 2nd Symposium on Large-Scale Digital Calculating Machinery. Harvard University Press, Cambridge, MA, 141-146.

[Levin, 2003] Levin, M.D. *Noise in gene expression as the source of non-genetic individuality in the chemotactic response of Escherichia coli.* FEBS Lett. Aug 28;550(1-3):135-8.

[Lobban and Sutton, 1973] Lobban, P.E., Sutton, C.A. *Enzymatic end-to-end joining of DNA molecules.* J. Mol. Biol., 453471.

[Lucas *et al.*, 2001] Lucas, P., Otis, C., Mercier, J.P., Turmel, M., Lemieux, C. *Rapid evolution of the DNA-binding site in LAGLIDADG homing endonucleases.* Nucleic Acids Res. Feb 15;29(4):960-9.

[Luria and Delbruck, 1943] Luria, S.E., Delbruck, M. *Mutations of bacteria from virus sensitivity to virus resistance.* Genetics 28:491.

[Luria, 1970] Luria, S.E. *The recognition of DNA in bacteria.* Sci Am. Jan;222(1):88-92.

[Ma *et al.*, 2004] Ma, H., Kumar, B., Ditges, U., Gunzer, F., Buer, J., Zeng, A. *An extended transcriptional regulatory network of Escherichia coli and analysis of its hierarchical structure and network motifs.* Nucleic Acids Res. Dec 16;32(22):6643-9.

[Macnab and Koshland *et al.*, 1972] Macnab, R.M., Koshland, D.E. *The gradient-sensing mechanism in bacterial chemotaxis.* Proc Natl Acad Sci USA. Sep;69(9):2509-12.

[McAdams and Arkin, 1997] McAdams H., Arkin, A. *Stochastic mechanisms in gene expression.* Proc Natl Acad Sci U S A. Feb 4;94(3):814-9.

[McAdams and Arkin, 1999] McAdams, H.H., Arkin, A. *It's a noisy business! Genetic regulation at the nanomolar scale.* Trends Genet. Feb;15(2):65-9.

[McDaniel and Weiss, 2005] McDaniel, R., Weiss, R. *Advances in synthetic biology: on the path from prototypes to applications* Current Opinion in Biotechnology. July. Vol. 16, 476-483.

[McMillen *et al.*, 2002] McMillen, D., Kopel N., Hasty, J., and Collins, J. *Synchronizing genetic relaxation oscillators by intercell signaling*, Proceedings of the National Academy of Science 99, 670-684.

[Meir *et al.*, 2002] Meir, E., Munro, E.M., Odell, G.M., Von Dassow, G. *Ingeneue: a versatile tool for reconstituting genetic networks, with examples from the segment polarity network.* J Exp Zool. Oct 15;294(3):216-51.

[Menabrea, 1842] Menabrea, L. F. *Sketch of the Analytical Engine* as translated with extensive explanations by Augusta, A..

[Messer, 2002] Messer, W. *The bacterial replication initiator DnaA. DnaA and oriC, the bacterial mode to initiate DNA replication.* FEMS Microbiol. Rev. 26, 355374.

['Mesoplasma'] *Mesoplasma florum Sequencing Project.* Broad Institute, `http://www.broad.mit.edu/annotation/microbes/mesoplasma_florum`. Accessed December 2005.

[Murray and Hunt, 1993] Murray, A. and Hunt, T. *The Cell Cycle: An Introduction* Oxford University Press, USA.

[Mushegian and Koonin, 1996] Mushegian, A. R. and Koonin, E. V. *A minimal gene set for cellular life derived by comparison of complete bacterial genomes.* Proc. Natl. Acad. Sci. USA 93: 10268-10273.

[NCIUB, 1985] Nomenclature Committee of the International Union of Biochemistry. *Nomenclature for incompletely specified bases in nucleic acid sequences*, Eur J Biochem 150: 1-5.

[Nelson and McClelland, 1989] Nelson, M., McClelland, M. *Effect of site-specific methylation on DNA modification methyltransferases and restriction endonucleases.* Nucleic Acids Res. 17 Suppl:389-415.

[Neumann, 1945] von Neumann, J. *First Draft of a Report on the EDVAC.* University of Pennsylvania, Report for the U.S. Army Ordnance Department.

[Nicholson *et al.*, 2000] Nicholson, W.L., Munakata, N., Horneck, G., Melosh, H.J., Setlow, P. *Resistance of Bacillus endospores to extreme terrestrial and extraterrestrial environments.* Microbiol Mol Biol Rev. Sep;64(3):548-72.

[Oberdoerffer *et al.*, 2003] Oberdoerffer P., Otipoby K.L., Maruyama M., Rajewsky K. *Unidirectional Cre-mediated genetic inversion in mice using the mutant loxP pair lox66/lox71.* Nucleic Acids Res. 15;31(22):e140.

[Orban *et al.*, 1992] Orban PC, Chui D, Marth JD. *Tissue- and site-specific DNA recombination in transgenic mice.* Proc Natl Acad Sci USA. Aug 1;89(15):6861-5.

[Ozbudak *et al.*, 2002] Ozbudak, E., Thattai, M., Kurtser, I., Grossman, A., van Oudenaarden, A. *Regulation of noise in the expression of a single gene.* Nat Genet. May;31(1):69-73.

[Pennisi, 2005] Pennisi, E. *Synthetic Biology Remakes Small Genomes.* Science, November, Vol. 310. no. 5749, 769-770.

[Prescott, 1994] Prescott, D.M. *The DNA of ciliated protozoa.* Microbiol Rev. 58(2), 233-267.

[Prescott *et al.*, 2001] Prescott, D.M., Ehrenfeucht, A., Rozenberg, G. *Molecular operations for DNA processing in hypotrichous ciliates.* European Journal of Protistology 37, 241-260.

[Ptashne *et al.*, 1976] Ptashne, M., Backman, K., Humayun, M.Z., Jeffrey, A., Maurer, R., Meyer, B., Sauer, R.T. *Autoregulation and function of a repressor in bacteriophage lambda.* Science. 1976 Oct 8;194(4261):156-61.

[Ptashne *et al.*, 1980] Ptashne, M., Jeffrey, A., Johnson, A.D., Maurer, R., Meyer, B.J., Pabo, C.O., Roberts, T.M., Sauer, R.T. *How the lambda repressor and cro work.* Cell. 1980 Jan;19(1):1-11.

[Ptashne, 2004] Ptashne, M. *A Genetic Switch: Phage Lambda Revisited* Cold Spring Harbor Laboratory Press, Third Edition.

[Podolsky, 1953] Podolsky, R. *Protein degradation in bacteria.* Arch Biochem Biophys. Aug;45(2):327-40.

[Pollard and Earnshaw, 2004] Pollard, T.D., Earnshaw, W.C. *Cell Biology* Saunders. November.

[Rao *et al.*, 2002] Rao, C.V., Wolf, D.M., Arkin, A.P. *Control, exploitation and tolerance of intracellular noise.* Nature. Nov 14;420(6912):231-7.

[Roberts *et al.*, 2003] Roberts, R.J., Vincze, T., Posfai, J., Macelis, D. *REBASE: restriction enzymes and methyltransferases.* Nucleic Acids Res. 1;31(1):418-20: with regard to the electronic resource at `http://rebase.neb.com/`.

[Roberts, 2005] Roberts, R.J. *How restriction enzymes became the workhorses of molecular biology.* PNAS, April 26, 102(17):5905-5908.

[Rosenberg *et al.*, 1978] Rosenberg, M., Court, D., Shimatake, H., Brady, C., Wulff, D.L. *The relationship between function and DNA sequence in an intercistronic regulatory region in phage lambda.* Nature. 1978 Mar 30;272(5652):414-23

[Rozen and Skaletsky, 2000] Rozen, S. and Skaletsky, H.J. *Primer3 on the WWW for general users and for biologist programmers.* Bioinformatics Methods and Protocols: Methods in Molecular Biology. Humana Press, Totowa, NJ, pp 365-386.

[Sadowski, 1986] Sadowski, P. *Site-Specific Recombinases: Changing Partners and Doing the Twist.* Journal of Bacteriology, 165(2):341-47.

[Sadowski, 2003] Sadowski, P. *The Flp double cross system a simple efficient procedure for cloning DNA fragments.* BMC Biotechnol. Jul 18;3:9.

[Salgado *et al.*, 2004] Salgado, H., Gama-Castro, S., Martinez-Antonio, A., Diaz-Peredo, E., Sanchez-Solano, F., Peralta-Gil, M., Garcia-Alonso, D., Jimenez-Jacinto, V., Santos-Zavaleta, A., Bonavides-Martinez, C., Collado-Vides, J. *RegulonDB (version 4.0): transcriptional regulation, operon organization and growth conditions in Escherichia coli K-12.* Nucleic Acids Res. Jan 1;32

[Sambrook and Russel *et al.*, 2001] Sambrook, J. Russell, D.W. *Molecular Cloning: a Laboratory Manual.* Third Edition, New York, Cold Spring Harbor Laboratory Press.

[Schatz, 1999] Schatz, D.G. *Transposition mediated by RAG1 and RAG2 and the evolution of the adaptive immune system.* Immunol Res. 1999;19(2-3):169-82.

[Schwikowski *et al.*, 2000] Schwikowski B., Uetz P., Fields S. *A network of protein-protein interactions in yeast.* Nat Biotechnol. 18(12):1257-61.

[Segall *et al.*, 1986] Segall, J.E., Block, S.M., Berg, H.C. *Temporal comparisons in bacterial chemotaxis.* Proc Natl Acad Sci USA. Dec;83(23):8987-91.

[Seligman *et al.*, 2002] Seligman, L.M., Chisholm, K.M., Chevalier, B.S., Chadsey, M.S., Edwards, S.T., Savage, J.H., Veillet, A.L. *Mutations altering the cleavage specificity of a homing endonuclease.* Nucleic Acids Research, Vol. 30, No. 17 3870-3879.

[Shapiro, 2002] Shapiro, J.A. *Genome Organization and Reorganization in Evolution: Formatting for Computation and Function.* Ann. NY Acad Sci 981, 111-134.

[Shea and Ackers, 1985] Shea, M.A., Ackers, G.K. *The OR control system of bacteriophage lambda. A physical-chemical model for gene regulation.* J Mol Biol. Jan 20;181(2):211-30.

[Shen-Orr *et al.*, 2002] Shen-Orr, S., Milo, R., Mangan, S., Alon, U. *Network motifs in the transcriptional regulation network of Escherichia coli.* Nat Genet. May;31(1):64-8.

[Shimizu-Sato *et al.*, 2002] Shimizu-Sato, S., Huq, E., Tepperman, J.M., Quail, P.H. *A light-switchable gene promoter system.* Nat Biotechnol. Oct;20(10):1041-4.

[Simpson *et al.*, 2004] Simpson, M.L., Cox, C.D., Peterson, G.D., Sayler, G.S. *Engineering in the biological substrate: information processing in genetic circuits* Proceedings of the IEEE, Volume 92, Issue 5, 848-863.

[Smith and Wilcox, 1970] Smith, H.O., Wilcox, K.W. *A restriction enzyme from Hemophilus influenzae. I.* J. Mol. Biol. 51, 379-391.

[Smith *et al.*, 1977] Smith, T.F., Sadler, J.R. and Goad, W. *Statistical-mechanical modelling of a regulatory protein: the Lactose repressor.* Math. Biosci. 36, 61-86.

[Spudich and Koshland, 1976] Spudich, J.L., Koshland, D.E. *Non-genetic individuality: chance in the single cell.* Aug 5;262(5568):467-71.

[Stankevicius *et al.*, 1998] Stankevicius K., Lubys A., Timinskas A., Vaitkevicius D., Janulaitis A. *Cloning and analysis of the four genes coding for Bpu10I restriction-modification enzymes.* Nucleic Acids Res. 15;26(4):1084-91.

[Stark *et al.*, 1992] Stark, W.M., Boocock, M.R., Sherratt, D.J. *Catalysis by site-specific recombinases.* Trends Genet. 8:432-439.

[Stocker *et al.*, 2003] Stocker, J., Balluch, D., Gsell, M., Harms, H., Feliciano, J., Daunert, S., Malik, K. and Van Der Meer, J., R. *Development of a Set of Simple Bacterial Biosensors for Quantitative and Rapid Measurements of Arsenite and Arsenate in Potable Water.* Environ. Sci. Technol. , 37, 4743-4750.

[Tchuraev, 1991] Tchuraev, R.N. *A new method for the analysis of the dynamics of the molecular genetic control systems. I. Description of the method of generalized threshold models.* J Theor Biol. Jul 7;151(1):71-87.

[Thomas, 1978] Thomas, R. *Logical analysis of systems comprising feedback loops.* J Theor Biol. Aug 21;73(4):631-56.

[Turing, 1936] Alan Turing. *On Computable Numbers with an Application to the Entscheidungsproblem.* Proceedings of the London Math Society 2(42), pp. 173-198,

[Turner *et al.*, 2004] Turner, T.E., Schnell, S., Burrage, K. *Stochastic approaches for modelling in vivo reactions.* Comput Biol Chem. Jul;28(3):165-78.

[Tyndall, 1877] Tyndall, J. *Further researches on the department and vital persistence of putrefactive and infective organisms from a physical point of view.* Phil Trans R Soc. 167:149206.

[Vallejo *et al.*, 1994] Vallejo, A.N., Pogulis, R.J., Pease, L.R. *In vitro synthesis of novel genes: mutagenesis and recombination by PCR.* PCR Methods Appl. Dec;4(3):S123-30.

[Wall *et al.*, 2003] Wall, M., Hlavacek, W., Savageau, M. *Design principles for regulator gene expression in a repressible gene circuit.* J Mol Biol. Sep 26;332(4):861-76.

[Wall *et al.*, 2004] Wall, M., Hlavacek, W., Savageau, M. *Design of gene circuits: lessons from bacteria.* Nat Rev Genet. Jan;5(1):34-42.

[Webb and White, 2005] Webb, K., White, T. *UML as a cell and biochemistry modeling language.* Biosystems. Jun;80(3):283-302

[Weisberg and Landy, 1983] Weisberg, R., Landy, A. *Site specific recombination in phage.* Lambda II. Cold Spring Harbor Laboratory, Cold Spring Harbor, N.Y. 211-250.

[Weiss *et al.*, 2003] Weiss, R., Basu, S., Hooshangi, S., Kalmbach, Karig, D., Mehreja, R., and Netravali, I.. *Genetic circuit building blocks for cellular computation, communications, and signal processing* Natural Computing. Vol. 2, 47-84.

[Weiss and Basu, 2002] Ron Weiss and Subhyu Basu. *The device physics of cellular logic gates.* The First Workshop on NonSilicon Computing, pages 54-61.

[Winfree, 2000] Winfree, A.T. *The Geometry of Biological Time.* Springer Verlag, Second Edition.

[Yagil and Yagil, 1971] Yagil, G., Yagil, E. *On the relation between effector concentration and the rate of induced enzyme synthesis.* Biophys J. Jan;11(1):11-27.

[Yanisch-Perron *et al.*, 1985] Yanisch-Perron, C., Vieira, J., Messing, J. *Improved M13 phage cloning vectors and host strains: nucleotide sequences of the M13mp18 and pUC19 vectors.* Gene 33, 103-119.

[Yokobayashi *et al.*, 2002] Yokobayashi, Y., Weiss, R., Arnold, F. *Directed evolution of a genetic circuit.* Proc Natl Acad Sci USA. Dec 24;99(26):16587-91.