Imperial College London

Department of Life Sciences

Pattern Generation and Recognition in Synthetic Biology

Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy (PhD)

by

Georg K. A. Wachter

Supervisors:
Prof. Robert Endres
Prof. Mark Isalan
Prof. Ron Weiss

October 9, 2023

Wordcount: ca. 36,000

Would it save you a lot of time if I just gave up and went mad now?

- Douglas Adams

Statement of Originality

I hereby declare that this thesis, including the results therein, are my own work, unless explicitly stated otherwise. Other work is referenced and attributed appropriately.

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgements

As I sit here, writing the final words of my PhD thesis after having spent the vast majority of my life in education, I cannot help but reflect on how I ended up where I am today, and how grateful I am to so many people that have shared this journey with me so far. My first thanks will go to the many educators for not stifling, but encouraging my curiosity. I cannot imagine the patience it must have taken to deal with the energetic primary school kid who would crawl under tables, and try to find "a better and more efficient way" to trace the letter "a" on the blackboard; or the high school kid who would - just to make a point - argue that "Fire displays most signs of life that you just told us about Professor! How can fire be more alive than a virus??". If it had not been for my biology teacher entertaining those thoughts, I probably would not be sitting here today.

When mentioning my educators, three stand out above all else: My supervisors, Robert Endres, Mark Isalan, and Ron Weiss. Mark and Robert - thank you for your inspiring lectures during my undergraduate days. Your lecture series in synthetic biology and systems biology were truly the highlight of my taught education, and were the biggest reason I ultimately did my PhD in this field. I also want to thank each of you for the trust and independence you have given me during my research projects as an undergraduate.

Mark, my time in your lab building conditional terminators to sense miRNA was my first independent lab project, and I had the time of my life. This project after my second year also marked a turning point for me. In my first year of university, I finished in the bottom 10% of the cohort, and was questioning my academic and scientific ability. I will never forget one particular evening in the lab: Martina and I were working late (probably later than you would have wanted us to Mark - and definitely later than the security guard wanted us to, since we had to hide from him in the toilets). I loved those evenings, but was worried that this time next year, my days doing independent research might be over. Suddenly I noticed that our exam results came via email. As I pulled them up, I realised that I had made it onto the Dean's list. Of course I was ecstatic with this achievement. But more than that - this marked the day where I felt that doing original research did not have to have an expiration date. I could continue to try and discover something new, or build something that no one else has ever conceived. Special thanks also goes to my lab mates: Martina (whom I doubt I will ever get rid of at this point) , Emily, Jure and Tong, without whom my experience would have been completely different.

Robert, my BSc thesis that I wrote with you was some of the most challenging research I have ever done. I remember several days simply sitting before a Fourier or Laplace transform, thinking how on earth this mathematical expression could ever yield some information that I could understand. During this research project, I learned two invaluable life lessons. The first one was the humility to start from absolute 0. I had no real previous experience in physics or mathematics. But with time and effort (oh yes, lots of effort...), I slowly made progress. I still frequently draw from this experience. Before this project, when I encountered a difficult problem, I would hold my head in anguish, exclaiming that I did not even know where to start. Now, I view a point of absolute 0 as something peculiar that warrants a smile. I wonder, once I start looking, where will the start be? My second life lesson was to trust my instincts. I distinctly remember going to your office, and wanting to switch my modelling approach; and I was expecting to just be told to stick with the original plan. However, the response I got was: "I have no real experience with partial delay differential equations... But if you think it could work, you should give it a go!" That second sentence has motivated several new approaches throughout this thesis, countless failed experiments, and even a few successful ones.

Ron, I came into your lab towards the tail-end of the Covid-19 pandemic. Before I arrived, my spirits were the lowest point during my PhD. I usually draw a lot of positive energy from the research I do, but my motivation was dwindling, and I was on the verge of burning out after weeks on end in a dark microscopy room searching for Turing patterns. When I arrived, your excitement for new discoveries was incredibly infectious, and could be felt in the lab every day. I hope that I can continue to carry a similar excitement for my work throughout my life. During my time at MIT, I also started to appreciate the freedom of responsibility. Very quickly, every day started to feel like a Saturday, and weeks like holidays. However, I also noticed, that I would spend long irregular hours in the lab without hesitation. Looking back, I think this freedom came from the trust and responsibility that I received from day 0. I also want to thank the entire Weiss lab for their help and guidance, as well as being a huge source of inspiration and joy while I was there. I want to especially thank Sarah, Kamila, Rashmi, Noreen, Andrew, Deepak, Winson and Akshay, as well as Peyton Randolph from the Liu lab. I particularly want to thank Charles and Jean, who were an absolute pleasure to work with on the neuromorphic project; I cannot wait to work with both of you again, and am excited to see how far we can take this project.

Several more dear friends have been a truly invaluable support network and sources of inspiration, both in the US and in the UK. If you count yourself as one of them and are

reading this, then you know that your name should be inserted here. I also want to thank the many friends I made (or where friendships grew stronger) during my time at Nucleate. I especially want to thank Kush here, who was an endless source of excitement and big dreams, and James, for bringing Nucleate to the UK in the first place.

Special thanks also goes to the funding agencies that enabled the research in this thesis: The Volkswagen Stiftung (LIFE: 93 065) for funding the experimental work on generating Turing patterns. The US National Institute of Health (NIH R21 grant CA257980) and the US Army Research Office (W911NF2120109) for funding the work on neuromorphic computing. I want to give particular thanks to Imperial College London for awarding me the Schrödinger scholarship, which is awarded to a single outstanding student in each department of the faculty of natural sciences every year. The freedom this scholarship provided was a fantastic opportunity, and allowed me to shape this PhD to an extent that would have otherwise not been possible.

As always, my greatest thanks goes to my family: Augustin, for always being a piece of home when I need it the most; my sister Thira, for making me laugh; my father Rainer, for motivating me to always keep pushing forward; and my mother Maria, for her care and kindness.

Abstract

Multicellular life requires both pattern formation and pattern recognition in order to form a functioning organism. In this thesis I set out to engineer both, in an attempt to harness these processes for synthetic biology, and to gain a deeper appreciation for biological complexity. I first attempt to engineer synthetic Turing patterns to generate diverse gene expression patterns in *E. coli*. I then set out to understand generated patterns using mathematical theory, especially under the consideration of growth. To advance gene expression pattern recognition, I build molecular neural networks in mammalian cells. To achieve this, I build molecular perceptrons, and chain them into two example neural networks; one forming a dual region classifier, the other a bandpass. Additionally, I lay the foundations for *in vivo* weight optimisation algorithms. While typically treated as separate issues, convergence of pattern generation and recognition could enable spatio-temporal control of morphogenesis, and eventually unlock macro-level design of biology.

Contents

1 Introduction: Importance of Patterns in Biology 10
- 1.1 Pattern Generation and Recognition: Two Sides of the Same Coin . . . 10
- 1.2 Pattern Generation . . . 11
  - 1.2.1 Spatial Pattern Generation in Biology . . . 11
  - 1.2.2 Analysing Turing Patterns . . . 15
  - 1.2.3 Turing Patterns and the Robustness Problem . . . 19
- 1.3 Pattern Recognition . . . 22
  - 1.3.1 Recognising Gene Expression Patterns in Biology . . . 22
  - 1.3.2 Neuromorphic Computing . . . 23
  - 1.3.3 Building a Molecular Neural Network . . . 28
- 1.4 Aims . . . 31

2 Pattern Generation: Engineering Synthetic Turing Patterns 34
- 2.1 Circuit Design . . . 34
- 2.2 Subcircuit Characterisation . . . 37
- 2.3 Characterisation of Diffusive Components . . . 42
- 2.4 Microscopy . . . 46
  - 2.4.1 Full Circuit . . . 46
  - 2.4.2 Control 1: Removing Diffusor Interactions . . . 49
  - 2.4.3 Control 2: Node A Deletion . . . 50
- 2.5 Considering Turing Instability and Growth . . . 52
- 2.6 Discussion & Conclusion: Engineering Turing Patterns . . . 60

3 Pattern Recognition: Neuromorphic Computing 63
- 3.1 Building Molecular Perceptrons . . . 63
- 3.2 Designing a Library of Biological Weights . . . 67
- 3.3 Including a Biological Bias . . . 70
- 3.4 Multilayer Molecular Perceptrons . . . 74
- 3.5 Updating Molecular Perceptron Weights *in vivo* . . . 79
  - 3.5.1 Introduction to prime editing . . . 79
  - 3.5.2 Design of a Weight Updating Algorithm . . . 82
  - 3.5.3 Implementation of a Weight Updating Algorithm . . . 84
- 3.6 Discussion & Conclusion: Neuromorphic Computing in Synthetic Biology . . . 90

4 Discussion and Conclusion 96

5 Methods 101
- 5.1 Dispersion Relation . . . 101
- 5.2 Dispersion Relation on Growing Domains . . . 104
- 5.3 Numerical Solving of Partial Differential Equations . . . 108
- 5.4 Transfection . . . 113
- 5.5 Flow Cytometry . . . 114
- 5.6 Flow Cytometry Analysis . . . 115
- 5.7 Gibson Assembly . . . 121
- 5.8 MoClo Assembly . . . 122
- 5.9 (e)pegRNA Assembly . . . 125

5.10 Electrotransformation . . . 127
5.11 Heat Shock Transformation . . . 129
5.12 Cell Strains Used . . . 130
5.13 Make Electrocompetent Cells . . . 130
5.14 Make Heat Shock Competent Cells . . . 132
5.15 Microscopy . . . 134
5.16 Liquid Culture Experiments . . . 135
5.17 Diffusion Characterisation . . . 138
5.18 Figure Design . . . 139
5.19 Parts Used . . . 139
5.20 Construction Primers . . . 143
5.21 Plasmids Used . . . 152

6 Reference list **173**

Appendix A Additional Liquid Culture Data for Circuit 3954 controls **182**

Appendix B Table of Abbreviations **183**

1

Introduction: Importance of Patterns in Biology

In this Chapter, I introduce biological patterning in the context of spatial pattern generation and gene expression pattern recognition. In addition to introducing the key technical concepts which form the foundation of this thesis, I also review the current state of the art to contextualise subsequent chapters. Finally, I lay out my aims for building gene circuits that can generate and recognise gene expression patterns.

1.1 Pattern Generation and Recognition: Two Sides of the Same Coin

For thousands of years, humans have been trying to make sense of life. Greek philosophers debated about its meaning, religions try to explain why it came to be, and scientists aim to find mechanistic understanding of life. What is this "special sauce" that makes us different from a rock floating through space? In his 1944 lecture series and book "What is Life?", Erwin Schrödinger proposes several answers to how life can exist. He does this from the perspective of a physicist of the 1940s, that is struggling to understand how life apparently violates the second law of thermodynamics, by being a self-maintaining area of low entropy. Before the discovery of DNA, he reaches the conclusion that living things must contain a carrier of information. He proposes the existence of an aperiodic crystal, that contains all the information to build an organism. This idea of self-organisation not only baffled Schrödinger, but other great biologists and theorists as well, such as legendary mathematician Alan Turing. Turing shifted the focus on emergent pattern formation - the process by which a single cell, containing the same information, splits into a complex organism via self-organisation.

Arguably, life could be seen as the ultimate emerging and self-replicating pattern. Pattern formation is ubiquitous in biology: From the sub-cellular level in bacteria, over initial symmetry breaking in development of multicellular organisms all the way to macro phenomena

such as finger or rib spacing. Without emergent patterns, life could not exist. In his seminal paper published in 1952, Alan Turing proposed a mathematical theory how two interacting chemicals - *morphogens* - would be enough to induce pattern formation via diffusion-driven instability. However, intuitively, we know that a self-replicating pattern by itself is not enough to constitute 'life', if it does not have any other processing capability, memory, or higher function.

This brings us to another hallmark of life: Pattern recognition. Living things excel at recognising and responding to various patterns. Again, this phenomenon occurs on every scale of life: Be it sub-cellular sensing and responding to gene expression patterns, to macro-level processing, such as a PhD student who attempts to make sense of a scientific paper. Indeed, humans have evolved to be outstanding pattern-recognition machines. In an attempt to improve computer programs, computer scientists have been working on neuromorphic computing: an artificial imitation of how the brain recognises and processes patterns. After around 80 years of research we now know that these attempts were successful - we encounter neuromorphic computing in the form of artificial neural networks every day. Neuromorphic computing helps us to diagnose diseases, changes how we interact with our phones via speech recognition, and (for all you know) could have helped to write this very introduction.

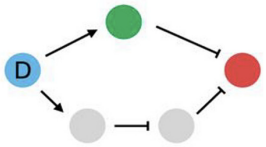
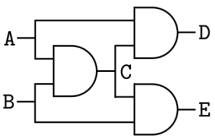
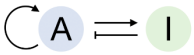
In living systems as we know them today, pattern generation and recognition are inextricably linked. Pattern generation requires some form of pattern recognition, to properly process gene expression patterns into a robust spatial structure. Additionally, pattern recognition mechanisms rely on intricate structure, that would not exist without a pattern generation mechanism producing the required layout. As such, these two processes exist as two sides of the same coin. Harnessing biological pattern generation or recognition separately would already lead to fantastic applications, such as self-organising biomaterials, growing replacement organs, or highly specific cell therapies. Mastery over both pattern generation and recognition would allow for macro level design of biology from the ground up.

1.2 Pattern Generation

1.2.1 Spatial Pattern Generation in Biology

Scientists have long attempted to demystify how a cell develops into a complex organism with distributed functions (Schrödinger, 1948). Over the years, several theories on how this can happen have been put forward. Some of those models, such as L-systems are simply algorithmic - for example how trees grow branches at certain points at certain angles. No information other than spacing to the previous branch is required, and then a new branch

	L-Systems	French Flag	Hierarchical	Turing
Type	Algorithmic	RD Class 1	RD Class 2	RD Class 3

Logic	`While growing:`
	`  wait timestep`
	`  split tip in 2`
				

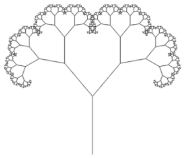
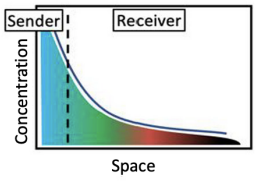
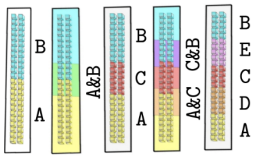
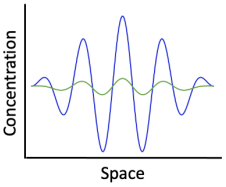
Pattern				

Figure 1: Examples for each reaction-diffusion (RD) Class shown in Table 1. The L-system logic is based on Lindenmayer (1968) (Prusinkiewicz and Lindenmayer, 1990; Bourke, 1991). The picture for the pattern was taken from Bourke (1991). The logic for the French Flag model is based on Basu et al. (2005). Circuit image and pattern image adapted from Oliver Huidobro et al. (2021). The logic for the hierarchical pattern is based on Boehm et al. (2018). The image for the hierarchical pattern is adapted from Boehm et al. (2018). The logic for the Turing pattern is based on Turing (1952).

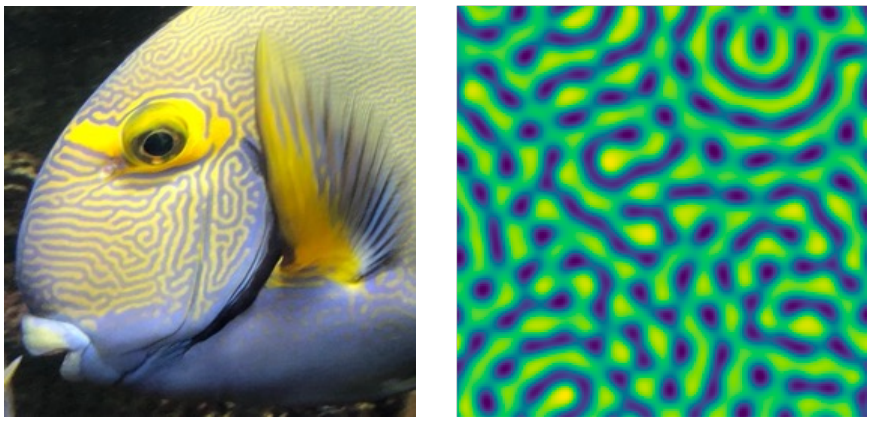
emerges at a certain angle (Lindenmayer, 1968). An example set of rules and an image of a pattern resulting from L-systems is shown in Figure 1. These rule-based models can also explain other simple structures appearing in biology, such as the zig-zag pattern in growing colonies of rod-shaped bacteria (Vassallo et al., 2019). One algorithmic pattern formation theory that can form more intricate patterns was proposed by Potts (1952). Here, cells spontaneously split into two populations of differential adhesion. One of these sub-populations adheres more strongly to other cells than the other. If all cells are allowed to move randomly, this will lead to cell sorting, where the strongly adhering cells are clumping together in the middle, and weakly adhering cells surround them, reminiscent of a blastocyst during embryogenesis (Alberts et al., 2008). Changing some parameters of the model can lead to more complex patterning such as spot or labyrinth formation (Graner and Glazier, 1992). This cell sorting mechanism has been demonstrated in mammalian cells, where two cell types with different adhesive properties were mixed at specified ratios. After allowing enough time for cells to move and sort, differently sized spots were observed, depending on the mixed ratio (Tordoff et al., 2021). In addition, some oscillatory phenomena can also produce spontaneous patterns, such as rings in growing *E. coli* colonies (Liu et al., 2011; Potvin-Trottier et al., 2016; Riglar et al., 2019).

Many other models assume that distant cells can communicate with each other via small diffusing molecules called morphogens, rather than only by direct contact. In general, a mor-

phogen is a signalling factor that can change the gene expression of a distant cell. These models can be sorted into three different categories. Firstly, models where cells only respond to morphogens, but do not produce them. The most well-known model in this category is the French flag model (Thompson, 1942; Wolpert, 1969; Basu et al., 2003). Cells respond to a morphogen gradient that gets secreted at a source, and then degrades. By sensing not only high concentration within the gradient versus low concentration, but also intermediate concentrations, cells are able to form different regions of gene expression (Figure 1). Furthermore, by adding more than one source, more complicated patterns, such as a Union Jack, could form (Kondo and Miura, 2010). Synthetic biologists have successfully engineered this patterning mechanism in *E. coli*. By placing a source of signalling molecule in the middle of a bacterial lawn, three distinct regions of gene expression formed in concentric circles around the source (Basu et al., 2005; Schaerli et al., 2014). An example of a French Flag model is shown in Figure 1. Without the production of more morphogens, the French flag model is limited by how many levels of concentration the biological system can differentiate between. For instance, if a biological system can differentiate between 3 levels of morphogen (low, medium and high concentration), this model could only ever set up three distinct stripes. If the scale of the system changes, e.g. it becomes longer, the stripes would just appear broader, rather than another stripe appearing. This excludes the French flag mechanism from being able to produce high frequency repeated patterns, unless these are explicitly coded within the concentration gradient recognition circuit. Even then, using a single morphogen is limited by biological noise, since these intermediate concentrations would have to be discernible for an arbitrary number of different regions of gene expression.

Hierarchical patterns are an extension of the French flag model, where one or more such distinct region of gene expression produces another morphogen (Thompson, 1942; Wolpert, 1969; Boehm et al., 2018). Including multiple morphogens removes the limitation where multiple closely spaced regions of gene expression need to form over a larger space. Through the inclusion of several produced morphogens, several repeat stripes can form, such as the ones during *Drosophila* development (Isalan et al., 2005; Wolpert, 2016). Hierarchical patterns have also been engineered from scratch in *E. coli*. By using n number of orthogonal morphogens, $2n$–1 stripes can form in 1D (Boehm et al., 2018). An example of an engineered hierarchical pattern is shown in Figure 1. While more scalable than a French flag model, hierarchical patterns still have several drawbacks. Firstly, the number of readily available orthogonal morphogens is limited (Tamsir et al., 2011; Racovita et al., 2022; Oliver Huidobro et al., 2021). Additionally, any kind of pattern that relies on a static source of morphogen

to set up a gradient is pushing the problem of initial symmetry breaking to an earlier time point.



(a) Eyestripe Surgeonfish (b) Turing Pattern

Figure 2: Comparison of an Eyestripe Surgeonfish (Figure 2a) and a Turing pattern simulation (Figure 2b) based on the circuit implemented in Section 2. I took the photo of the Eyestripe Surgeonfish in the *Haus des Meeres* in Vienna, Austria. As can be seen, the labyrinthine pattern of the simulation closely resembles the pattern on the fish.

In a seminal paper, (Turing, 1952) demonstrated that repetitive patterning could occur without the need for a pre-pattern or gradient, by using only two interacting morphogens. By having one slow diffusing activator, and one fast diffusing inhibitor, several intricate patterns such as labyrinths, stripes, and spots can form (Gierer and Meinhardt, 1972; Schnakenberg, 1979; Kondo and Miura, 2010). An example of what a Turing pattern looks like in 2D, as well as how it resembles patterns appearing in biology is shown in Figure 2. Turing patterns are also self-healing once the tissue is already developed and then perturbed. This was tested on the characteristic stripes of the zebrafish. After photobleaching a portion of the pattern, the pigment healed in a labyrinthine pattern, as predicted by a Turing model (Nakamasu et al., 2009). Further evidence that Turing patterns play an important role during development was shown by Raspopovic et al. (2014). Results indicated that digit formation in the mouse limb bud was due to Bmp-Sox9-Wnt signalling as part of a Turing network. Biological perturbations lead to developmental changes as predicted by a Turing model. Additionally, dissociated cells with eGFP tagged Sox9 form labyrinthine patterns *in vitro* (Raspopovic et al., 2014; Zuniga and Zeller, 2014).

Since the theory behind Turing patterns is so well defined, scientists and biological engineers have been trying to recreate them from scratch (Scholes and Isalan, 2017). Chemical Turing patterns were soon successfully built *in vitro* using a chlorite-iodide-malonic acid reaction (Castets et al., 1990). An initial study produced regularly spaced spots along a line,

which was then expanded to include more diverse 2D patterns such as stripes or hexagonally spaced spots (Castets et al., 1990; Ouyang and Swinney, 1991; Horváth et al., 2009). Stationary Turing patterns have also been built in chemical systems using proteins of the *E. coli* Min system (Glock et al., 2019).

Building Turing patterns in biology has however been a major challenge. The parameter space that supports Turing pattern formation is very narrow, and the inherent stochasticity of biology has been a longstanding obstacle to overcome (Diambra et al., 2015). One study produced Turing patterns in human cells using mouse Nodal and Lefty (as well as the relevant missing receptors) (Sekine et al., 2018). Nodal and Lefty were previously indicated to produce Turing patterns in early development (Almuedo-Castillo et al., 2018). The slow diffusing Nodal was engineered to produce more of Nodal as well as Lefty (slow diffusing activator), while Lefty inhibited the production of Nodal. This produced regularly spaced labyrinthine patterns, as predicted by a Turing model (Sekine et al., 2018). Another study engineered stochastic Turing patterns in *E. coli* from the ground up with signalling molecules not implicated in any kind of Turing pattern formation. While biological noise prohibited the regular spacing seen in chemical systems, slightly irregular spots formed on a bacterial lawn (Karig et al., 2018). This, in conjunction with other mathematical studies, indicates that noise (as well as other biological features) may actually aid pattern formation in some cases (Maini et al., 2012*a*,*b*).

A non-exhaustive list of different patterning mechanisms can be found in Table 1. The models described above are split into four different categories of reaction-diffusion (RD) mechanisms: Class 0 RD models do not use morphogens for patterning. Instead, they rely on algorithmic rules or mechanical properties. Class 1 RD systems do not need to produce any morphogens to function. The cells only need to recognise and respond to some sort of pre-pattern, such as a morphogen gradient. Class 2 RD systems require a pre-pattern such as a gradient, but are able to produce more complex patterning by producing at least one morphogen themselves. Class 3 RD systems do not require any pre-pattern, and produce and respond to multiple diffusing molecules. They can spontaneously form emerging patterns from a uniform field. In this thesis, the main patterning strategy is the Turing mechanism. One example mechanism for each of these classes is shown in Figure 1.

1.2.2 Analysing Turing Patterns

Turing patterns, as first proposed by Turing (1952) and expanded upon by Gierer and Meinhardt (1972), are composed of one slow diffusing activator, and one fast diffusing inhibitor. The activator produces more of itself, as well as more inhibitor, while the inhibitor

Name	RD Class	Theory Source	Engineering Medium	Engineering Source
Fractal	0	Lindenmayer (1968)	Biology	Vassallo et al. (2019)
Cellular Potts	0	Potts (1952); Graner and Glazier (1992)	Biology	Tordoff et al. (2021)
Surface minimisation	0	N/A	Mechanics	Hayashi and Carthew (2004)
Cell state arrest	0	N/A	Biology	Potvin-Trottier et al. (2016)
Auxin Flow	1	Grieneisen et al. (2007)	N/A	N/A
French Flag	1	Wolpert (1969)	Biology	Basu et al. (2005); Schaerli et al. (2014)
Hierarchical	2	Wolpert (1969)	Biology	Isalan et al. (2005); Boehm et al. (2018)
Spatial Oscillator	3	Murray (2002*a*)	Biology	Danino et al. (2010)
Turing (Classical)	3	Turing (1952); Gierer and Meinhardt (1972)	Chemistry	Castets et al. (1990); Ouyang and Swinney (1991); Horváth et al. (2009); Glock et al. (2019)
Tuing (Stochastic)	3	Maini et al. (2012*a*)	Biology	Karig et al. (2018)
Turing (Growth induced)	3	Madzvamuse et al. (2010); Van Gorder et al. (2021)	Chemistry	Konow et al. (2019)
Clock and Wavefront	3	Baker et al. (2006)	Biology	Liu et al. (2011)
PORD	3	Cotterell et al. (2015)	N/A	N/A

Table 1: A non-exhaustive list of different patterning mechanisms, including theory papers and engineering studies. RD classes are defined as follows: Class 0 systems have no diffusors and are not RD systems (e.g. algorithmic patterns). Class 1 systems pattern because of a single diffusor that is applied exogeneously. Class 2 patterns either have multiple diffusors or do not require a pre-pattern to form. Class 3 RD systems have multiple diffusors and do not require a pre-pattern to form.

stops production of the activator. A classic two-node network topology is illustrated in Figure 3, and can be described by Equation 1. How this leads to stationary pattern formation is shown in Figure 4. When a uniform field is slightly perturbed, the activator stays localised at a high concentration, and produces more of itself, as well as more of the inhibitor. The inhibitor diffuses more quickly, and spreads around the activator (Figure 4a). This creates one region where the activator is higher than the inhibitor, and a neighbouring region, where the inhibitor is higher than the activator. In the activator dominant region, more of activator and inhibitor are produced. In the inhibitor dominated region, the activator is depleted more quickly, leading to a region of low activator (Figure 4b). Because less activator is present, this also leads to a local reduction in inhibitor in the inhibitor dominated region. This allows for self-activation of the activator adjacent to the inhibitor dominated region (Figure 4c).

$$\begin{aligned} \underbrace{\frac{\partial A}{\partial t}}_{\text{Rate of change}} &= \underbrace{F_A(A,I)}_{\text{Reaction}} - \underbrace{\mu_A A}_{\text{Degradation}} + \underbrace{D_A \nabla^2 A}_{\text{Diffusion}} \\ \underbrace{\frac{\partial I}{\partial t}}_{\text{Rate of change}} &= \underbrace{F_I(A)}_{\text{Reaction}} - \underbrace{\mu_I I}_{\text{Degradation}} + \underbrace{D_I \nabla^2 I}_{\text{Diffusion}} \end{aligned} \tag{1}$$

Since most Turing patterns can be modelled with simple partial differential equations

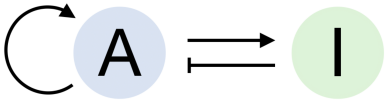
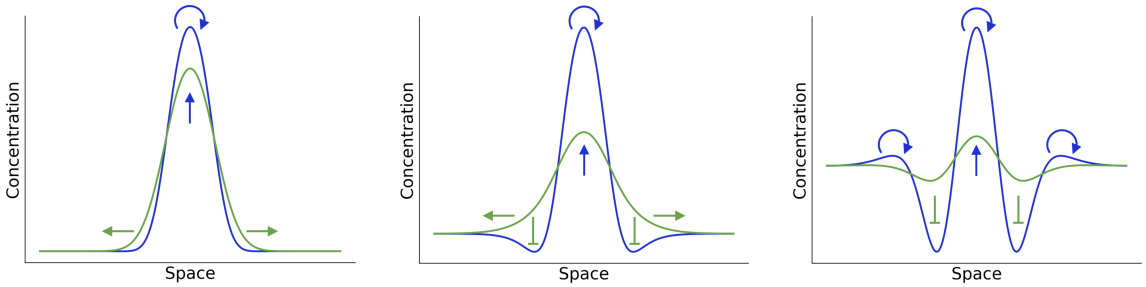


Figure 3: Graphical representation of a classical Turing topology (Turing, 1952; Gierer and Meinhardt, 1972).



(a) After 250 time steps (b) After 1500 time steps (c) After 15000 time steps

Figure 4: Evolution of a spatial perturbation over time in a Turing system. The blue and green lines represent the slow diffusing activator and the fast diffusing inhibitor respectively. The system is modelled after the original equations presented by Turing (1952). ($\frac{\partial X}{\partial t} = D_X \nabla X + 5X - 6Y + 1$ and $\frac{\partial Y}{\partial t} = D_Y \nabla Y + 6X - 7Y + 1$. The diffusion constants are set so $\frac{D_X}{D_Y} = 0.01$. Periodic boundary conditions are used. Time and space are arbitrary.)

(PDEs), they can be studied using analytical approaches, such as linear stability analysis and dispersion relation (Murray, 2002*b*,*a*; Strogatz, 2000). Using these methods, one can determine whether a set of parameters and equations can give rise to Turing patterns and what the characteristic wavelength is (how activator peaks are spaced). Analytical approaches are computationally far less expensive than simulating the patterning process numerically, allowing for higher throughput screens of parameter space. High-throughput analysis is necessary, because while many circuit topologies are able to produce Turing patterns, only very few parameter sets do so (Scholes et al., 2019).

In order to solve the dispersion relation for any simple reaction diffusion system, the simple 2 equation system in Equation 1 needs to be generalised. This can easily be done by rewriting the reaction-diffusion (RD) system as shown in Equation 2. S_i is the species to be modelled, F_i the function that describes how species S_i is regulated, S_j is any species that species S_i is dependent on, and D_i is the diffusion coefficient for S_i.

$$\frac{\partial S_i}{\partial t} = F_i[S_j(x,t)] + D_i \nabla^2 S_i \tag{2}$$

By applying a perturbation p to the steady state of this system, one can study whether the system will form a stable spatial pattern. This can be done by solving the characteristic equation shown in Equation 3. J^*_{ij} is the Jacobian matrix of a system of equations around a

steady state, k is the wavenumber (an indicator of periodicity of a spatial pattern), λ is the eigenvalue of the system, and I the identity matrix. The largest real eigenvalue measures the stability of a system. Equation 4 shows how a small perturbation p will evolve over time, where time t and the eigenvalue λ are the exponents, and v is a vectorised ODE system. If $\lambda < 0$, the system is stable around a certain steady state, and will return to it if it is subjected to small perturbations. If $\lambda > 0$, the system is unstable, and will diverge from a certain steady state. When screening for diffusion driven instability - the hallmark of a Turing pattern - the system must be stable without diffusion, and be unstable once diffusion is introduced. This becomes readily apparent when plotting the eigenvalue λ as a function of the wavenumber k. Since k is a measure proportional to the frequency, as shown in Equation 5, the plots can be easily interpreted.

$$det(J^*_{ij} - D_i k^2 - \lambda I) = 0 \tag{3}$$

$$p = e^{\lambda t} v \tag{4}$$

$$k = \frac{n\pi}{L}, n \in \{0, \mathbb{N}\} \tag{5}$$

For Turing (or diffusion driven) instability to occur, the system must be stable without diffusion - so $\lambda < 0$ at $k = 0$. The system must then be unstable ($\lambda > 0$) for some $0 < k < \infty$. For a characteristic wavelength to emerge, the system must then return to stability as $\lim_{k\to\infty}$. If the system is unstable at $k = 0$, but a wavenumber exists where the system is more unstable ($\lambda(k = 0) < \lambda(0 < k < \infty)$), the system will still form a stable spatial pattern (Manna et al., 2021). Due to the properties of the wavenumber shown in Equation 5, the wavelength of the emerging pattern can be calculated from the wavenumber at which the maximum real λ is greatest (in a finite system with closed boundary conditions). If a point exists for any finite wavenumber $k > 0$, this point is called a "dispersion peak". If the eigenvalue has an imaginary part, the pattern oscillates in the form of a travelling wave. The frequency of oscillation can also be calculated from Equation 5 (Murray, 2002*a*; Strogatz, 2000). A derivation of Equations 3 and 5 are shown in 5.1, along with a more detailed mathematical explanation.

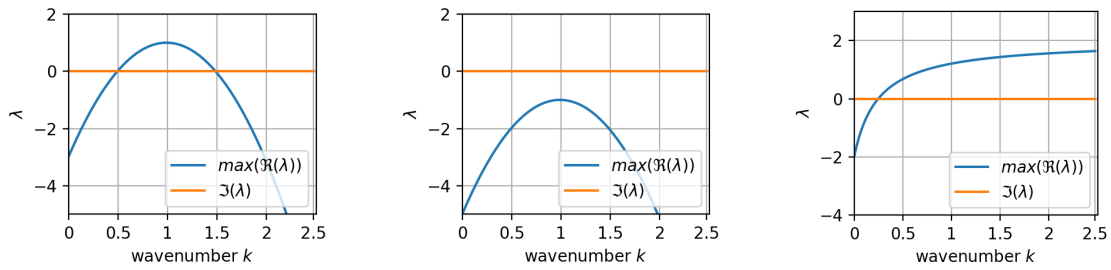
Intuitively, this concept can be understood in the following way: Imagine a well-mixed system, where all points have the same concentration. If all points are displaced (a perturbation is applied) with the same amplitude, diffusion will have no effect, since diffusion only cares about differences of concentration in space. Changing the wavenumber k is equivalent to now not displacing all points on the line in the same way, but by picking a specific *cos*

wave, and applying it as a perturbation. Now there are differences in concentration between points on the line, and diffusion has an effect. If there is a *cos* wave with a specific frequency that makes the system the most unstable, the system will now diverge from the steady state more quickly than with any other *cos* wave. This means the frequency of that wave will dominate the emerging pattern, and appear as the characteristic wavelength of the pattern. Figure 5 shows examples of dispersion plots that indicate pattern formation (or lack thereof) according to classical Turing theory.

1.2.3 Turing Patterns and the Robustness Problem

While many patterns appearing in nature resemble the typical spots, stripes and labyrinths created by Turing patterns, mathematical analysis reveals that - by conventional understanding - Turing patterns should by no means be so ubiquitous in nature. Turing patterns have very low parametric robustness; only very few parameter combinations give rise to Turing patterns (Maini et al., 2012*a*; Scholes et al., 2019). Over the past 20 years, a lot of mathematical modelling has been done to explain this disparity between the lack of parametric robustness and the apparent readiness at which Turing patterns appear in nature. Synthetic biologists aim to distil design principles from these studies, in order to build Turing patterns from scratch (Scholes and Isalan, 2017).

One strategy to explain this disparity focuses on finding general rules in the network, while continuing to use a standard PDE modelling framework. This approach reveals that adding an extra (non-diffusing) node to a Turing network (3 node network instead of the classical 2 node topology shown in Section 1.2.2) enables the diffusors to have equal diffusivity (Klika et al., 2012). Normally, differential diffusivity is one of the main drivers of Turing pattern formation, and most available diffusors available are predicted to diffuse at roughly equal speeds (Kondo and Miura, 2010; Oliver Huidobro et al., 2021). Another study examines a full network atlas of all 3 node networks with 2 diffusing nodes. This reveals that some networks are more lenient when it comes to parametric robustness. Furthermore, including redundant links between networks increases their topological robustness. This means if some interactions between nodes fail, the resulting network is still able to produce Turing patterns (Scholes et al., 2019). Interactions between nodes in synthetic gene circuits still fail in unpredictable ways, even when their components behave as intended in isolation (Espah Borujeni et al., 2020). Thus, increased topological robustness is immensely beneficial when attempting to build Turing patterns from scratch, as the actual resulting topology can still produce Turing patterns. Another study reveals that having steeper activation functions between nodes - higher cooperativity/Hill coefficient - leads to a more robust parameter space (Diambra et al.,



(a) Dispersion peak > 0: Turing pattern

(b) Dispersion peak < 0: No pattern[1]

(c) No Dispersion Peak: No pattern

Figure 5: Example dispersion plots. 5a Turing unstable, giving rise to stationary patterns, whereas cases in 5b and 5c do not. The blue line is the maximum real part of the eigenvalue λ at a specific wavenumber k, and indicates stability. The orange line is the imaginary part of the eigenvalue.

2015). Apart from using transcription factors that inherently have higher cooperativity, this can be purposefully be engineered by including positive feedback loops, or adding additional operator elements (Brophy and Voigt, 2014; Meyer et al., 2019; Rizik et al., 2022).

Another mathematical concept is including time delays when modelling Turing patterns. Time delays naturally occur in biology, such as by ribosome stalling during translation (Rodnina, 2016). These effects are usually neglected when modelling synthetic gene circuits (Elowitz and Leibler, 2000; Andrianantoandro et al., 2006; Kærn and Weiss, 2006). However, time delays are shown to impact pattern formation (Maini et al., 2012*a*). Some studies suggest that they may shift the parameter space of a Turing network, and can thereby hinder pattern formation (Yi et al., 2017). Other studies show that time delays can induce pattern formation, and increase parameter space (Forde, 2005; Sen et al., 2009; Zhang and Zang, 2014; Maini et al., 2012*a*; Puzyrev et al., 2013). This strategy has successfully been used to model plankton biofilm formation (Tian and Zhang, 2013). However, including time delays in modelling significantly complicates stability analysis. The characteristic equation that describes stability of time delay PDEs is $det\big(J_{ij}(k) + J_{ij\tau_n}(k)e^{-\lambda\tau_n} - \lambda I\big) = 0$. This is a transcendental equation (a quasi-polynomial), and thus cannot be solved for λ analytically (Lakshmanan and Senthilkumar, 2010).

Growth and curvature of the domain is also often neglected when modelling Turing patterns, but is implicated both in what shape the pattern will form, as well as its robustness (Plaza et al., 2004; Maini et al., 2012*a*,*b*). By including domain growth as well as dilution of the chemicals on the growing field due to conservation of mass, parametric robustness is

[1]While this results in no patterning in a classical (deterministic) Turing system, stochastic Turing theory suggests that this might still form more irregular Turing-like patterns. See Sections 1.2.3 and 2.5.

shown to increase. Additionally, accounting for growth also allows some network topologies to form Turing patterns that otherwise do not. As with time delays, including growth in the modelling strategy impedes analytical analysis. Analytical analysis is only possible when assuming isotropic, exponential and dilational growth (Madzvamuse et al., 2010). Even with these assumptions, such analysis can still break down in some cases, and should only be used as a rough indicator, before moving to more robust numerical predictions (Klika and Gaffney, 2017). More expensive numerical analysis can be used to model more biologically relevant cases of growth (e.g. transient or non-isotropic growth), and also show how patterning is affected on 2D and 3D surfaces (Van Gorder et al., 2021). In addition to mathematical analyses, chemical Turing systems have also been studied on a growing domain. A chemical reaction was inhibited via light on the outside of a circular disk, and the area where the reaction is not halted was expanded over time. This, in addition to mathematical modelling, revealed that the growth rate could also impact the shape of a pattern. Slow growth rates were likely to produce rings that were added on the inside of the disk as it grew (inner ring addition), while fast growth rates produced rings on the outside (outer ring addition). Intermediate growth rates formed labyrinthine structures (Konow et al., 2019).

Counterintuitively, the stochasticity of biological systems may actually improve the chances of producing Turing patterns (Butler and Goldenfeld, 2009; Biancalani et al., 2010). This is because of the false assumption that when a spatial system is perturbed from a stable steady state, the system returns to the steady state monotonically. However, this is not the case if the dispersion relation shows a peak for a characteristic wavelength - even if that peak is slightly below zero, and would not result in pattern formation in a deterministic system (Biancalani et al., 2017). Unlike their deterministic counterpart, stochastic Turing patterns also do not require large differential diffusivity between the activator and the inhibitor (Butler and Goldenfeld, 2011; Biancalani et al., 2017). While stochasticity seems to solve both the robustness problem, and makes way of the requirement for large differential diffusivity, the emerging patterns are not as ordered as deterministic Turing theory. Stochastic Turing patterns have been engineered from the ground up in *E. coli* using quorum sensing molecules and a synthetic circuit. The resulting patterns appeared as slightly irregular spots. Spectral analysis showed that there was a dominant characteristic wavelength. Further modelling also revealed that the pattern would not form with deterministic Turing theory, but stochastic simulations predicted a pattern to form (Karig et al., 2018). To date, deterministic Turing patterns have not been engineered from first principles with components that are not already implicated in patterning.

Inputs					Output
miR-21	miR-17-30a	miR-141	miR-142(3p)	miR-146a	hBax
1	1	0	0	0	1
all other miRNA combinations					0

Table 2: Truth table for a Boolean cancer classifier circuit, as described by Xie et al. (2011). A protein inducing apoptosis (*hBax*), is only expressed, if a miRNA profile specific to HeLa cells is present. All other miRNA combinations do not lead to *hBax* expression, and the cells survive.

1.3 Pattern Recognition

1.3.1 Recognising Gene Expression Patterns in Biology

Similarly to pattern generation, pattern recognition is ubiquitous in biology. From molecular signalling cascades to recognise DNA damage to image processing in the human brain, recognising and responding to patterns is what gives organised living matter function (Purvis and Lahav, 2013). Additionally, there are several cases where pattern formation and recognition are functionally linked. For example, the French flag model relies on a pre-patterned gradient; the resulting gene expression pattern then forms based on a pattern recognition mechanism. Different genes are expressed depending on different levels of a diffusing morphogen (Wolpert, 1969; Basu et al., 2003; Isalan et al., 2005; Basu et al., 2005; Schaerli et al., 2014; Santos-Moreno and Schaerli, 2019).

Apart from aiding to generate spatial patterns, pattern recognition mechanisms in synthetic biology can be engineered for numerous other applications such as targeted cell therapies (Kitada et al., 2018). In synthetic biology, curent gene expression pattern recognition algorithms are usually encoded in Boolean truth tables. When a combination of certain biomarkers (such as miRNAs) are up- or down-regulated, the desired output is achieved (Xie et al., 2011; Nielsen et al., 2016). One example application of such a classifier circuit is to differentiate between cancer cells (HeLa) and healthy cells (HEK) based on their respective miRNA profiles. This was done by finding 5 miRNAs that are significantly up- or down-regulated between those two cell types. By activating expression of *hBax* - a protein that induces apoptosis - cancer cells were instructed to self-destruct by the genetic programme. A truth table for this programme is shown in Table 2 (Xie et al., 2011). While useful as a first-generation classifier, many applications require the sensing of gene expression dynamics, or require recognition where sensing simple ON-OFF behaviour is insufficient (Purvis and Lahav, 2013; Kitada et al., 2018; Zhao et al., 2023).

While relatively easy to understand and communicate, hard coding these pattern recognition programs into Boolean circuits has a few drawbacks. Firstly, the network architecture

is bespoke to the recognition task at hand. This is not the case for other general pattern recognition frameworks, such as artificial neural networks (Abiodun et al., 2019). Boolean pattern recognition strategies require meticulous design, assembly, testing and improving of the circuit with every new recognition target (Meng and Ellis, 2020). Moreover, many signals cannot be classified as simply being high or low - biology operates in an analogue world, and intermediate concentrations can carry important information (Sauro and Kim, 2013; Purvis and Lahav, 2013; Vohradsky, 2001). Additionally, synthetic biology is still limited by the number of orthogonal parts that can be used in biological computation (Meyer et al., 2019; Oliver Huidobro et al., 2021; DiAndreth et al., 2022). Some studies have distributed these parts between different bacterial colonies to expand what computation is possible (Al-Radhawi et al., 2020; Tamsir et al., 2011). Using analogue computation strategies can decrease the number of parts needed for complex computational tasks (Sauro and Kim, 2013; Daniel et al., 2013; Rizik et al., 2022). In this thesis, gene expression pattern recognition strategies are implemented based on neuromorphic computing - an analogue computation framework loosely based on the human brain.

1.3.2 Neuromorphic Computing

The idea of neuromorphic computing - computation that is inspired by the brain - started in the 1940s. Warren McCulloch and Walter Pitts published a seminal paper in 1943 that proposes a mathematical theory of how a single neuron behaves and links with other neurons (McCulloch and Pitts, 1943). In 1958, their theory would be implemented by Frank Rosenblatt and become known as a *Perceptron* (Rosenblatt, 1958). The term neuromorphic computing was coined later, by Carver Mead in the late 1980s, and refers to brain-inspired hardware that deals with analogue signals (Mead, 1990). Neuromorphic computers are usually contrasted with von Neumann computers - where the first deals with analogue signals and logic, while the latter deals with digital (or binary/Boolean) signals and logic. Unlike von Neumann computers, neuromorphic computers co-localise their memory and computing elements. Neuromorphic computers have several advantages over digital computers. Firstly, they can better account for stochasticity in the system (Schuman et al., 2022). These advantages of neuromoprhic computers have inspired computer scientists to emulate analog computation on von Neumann hardware, such as in the artificial neural networks used today. However, even in computer science, analog computers are making a comeback because of the inefficiencies of analog emulation in a digital environment (Ulmann, 2022). The resistance to noise in neoromorphic computers is also critical in biological systems, which are inherently stochastic; especially on a single cell level (McAdams and Arkin, 1999). In fact, a molecu-

lar neural network using DNA has been shown to be able to correctly classify hand-written digits, even when 30% of the inputs were incorrect when compared to the reference image (Cherry and Qian, 2018). Secondly, neuromorphic computers are resistant to component failure (Mead, 1990; Hajnal et al., 1993). Resistance to component failure is again highly beneficial in synthetic biology, since well-characterised components and even whole gates have been shown to fail in Boolean logic circuits (Espah Borujeni et al., 2020). Furthermore, neuromorphic circuits use exponentially less parts as the complexity of the system grows, mainly due to not discarding analogue information (Wegener, 1991; Qian et al., 2011). Finally, neuromorphic computers follow the same architecture as most modern pattern recognition algorithms (Abiodun et al., 2019; Emmert-Streib et al., 2020; Schuman et al., 2022). Implementing such algorithms in biological systems could greatly enhance our ability to engineer cells that recognise and respond to their environment (Xie et al., 2011; McNerney et al., 2021; Zhao et al., 2023). The most prominently employed algorithm to recognise and categorise different spatiotemporal patterns right now is a multi-layer perceptron, more commonly referred to as an artificial neural network (Schuman et al., 2022). Figure 6 shows a simplified schematic of a biological neuron. In this simplified model, the functions of a neuron can be split into four components (Banoula, 2023):

1. The dendrites. Each dendrite receives information from other nerves, and carries these inputs to the cell body.

2. The synapses. Each synapse (the beginning of the dendrite) is connected either strongly or weakly with the input it is receiving, thereby assigning importance to its respective signal.

3. The cell body. The cell body then takes all of those inputs, and creates an action potential that is subsequently sent down the axon.

4. The axon terminals. The axon splits up into several different endings, and outputs the signal it receives from the cell body either to other neurons or to other cell types (e.g. muscle cells).

As seen in Figure 7, a perceptron (an artificial abstraction of a neuron) is structured in a similar way. It also takes several inputs, assigns relative importance to them and processes the inputs into a signal that is passed on. It can also be split into four distinct categories that functionally map to the biological neuron.

1. Inputs. Each input can be any number, that is passed to the perceptron for processing.

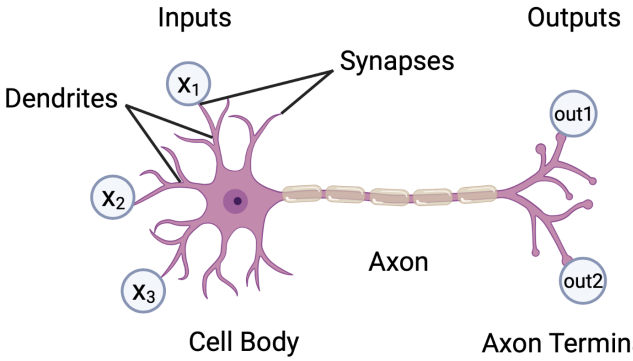


Figure 6: Schematic representation of a biological neuron. Inputs are received by the synapses on the dendrites, and are transferred to the cell body. The cell body then sends the signal down the axon. The output is then distributed to more cells via the axon terminals.

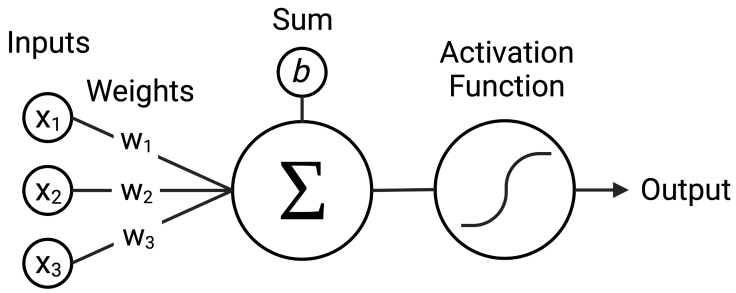


Figure 7: Schematic representation of an artificial neuron. Inputs are received, and multiplied by their respective weights. The weighted sum of the inputs is then calculated, added to a bias, and passed to an activation function. The result of that activation function is then used as an output, and can be connected to more artificial neurons, or used as an output directly.

2. Weights. Each input is multiplied by its respective weight, before it gets processed further. The weights assign relative importance to the input that is passed down for further processing. Note that weights can be positive or negative, and confer "memory" to the perceptron.

3. Summation. Once each input is multiplied with its relative weight, all results are added up and added to a bias.

4. Activation function. The result of the weighted summation is then passed to an arbitrary (differentiable) function. The result of the function is then either passed to the next connected artificial neuron, or output directly.

An artificial neuron can be easily described by Equation 6. This compact description encompasses all properties described above. y is the output of the perceptron. f is any arbitrary function. The bias, b, is a parameter that is assigned to the perceptron, and is a constant that gets added to the sum of weighted inputs once it is computed. The sum

adds all inputs x_i together once they have been multiplied with their respective weight w_i (Murphy, 2022).

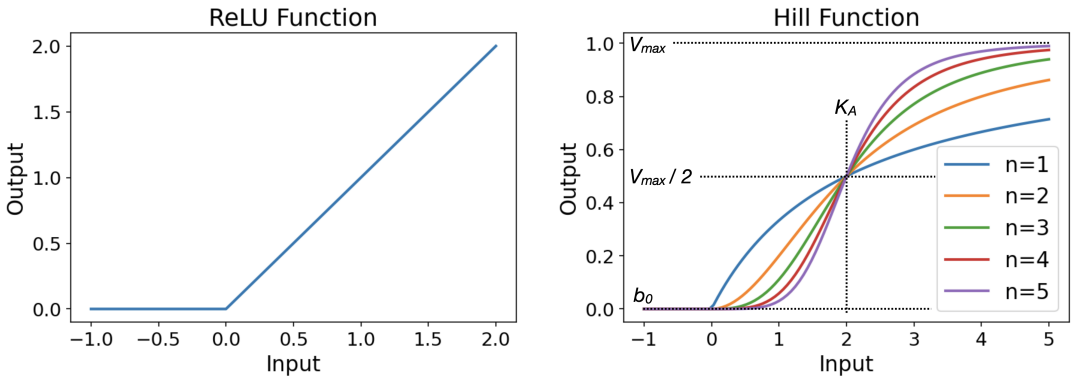
$$y = f(b + \sum_{i=0}^{m} w_i x_i) \tag{6}$$

Once the weighted sum is computed and the bias is added, the sum is passed to any (differentiable) function f. One of the most popular functions in computer science is the Rectified Linear Unit (ReLU) function. The ReLU function has reached such popularity for two main reasons: Firstly, the computation step is very easy. Secondly, the gradient is positive and constant for any input value above zero (unlike for other popular activation functions such as the unit step function or *tanh* function, where the gradient diminishes the further away the input is from 0). An easily obtainable gradient simplifies backpropagation, which is a popular weight optimisation strategy in computer science. Figure 8a shows the output of the ReLU activation function, shown in Equation 7. Since any arbitrary differentiable function can be used in a perceptron, it can also use a typical Hill equation. The Hill equation is a popular model for biochemical reactions, including gene expression. The output of the Hill function (Equation 8) is shown in Figure 8b. The sigmoidal Hill equation has some intuitive parameters. V_{max} is the maximum output the function can generate. K_A denotes the input at which half of V_{max} is reached. n is the cooperativity parameter, which is a measure of how steep the slope of the slope is at its midpoint. The Hill function typically also has a parameter b_0 for background activity (Murray, 2002*b*; Abiodun et al., 2019; Murphy, 2022).

$$y = \begin{cases} 0, & \text{if } x \leqslant 0 \\ x, & \text{otherwise} \end{cases} \tag{7}$$

$$y = \begin{cases} 0, & \text{if } x \leqslant 0 \\ b_0 + V_{max} \frac{x^n}{x^n + K_A^n}, & \text{otherwise} \end{cases} \tag{8}$$

As an example, a minimal perceptron with 2 inputs, one positive and one negative, and one output is illustrated in Figure 9. Following Equation 6, the resulting expanded equation for y is shown in Equation 9. Figure 10 shows how different activation functions, as well as tuning the weights and bias of a single perceptron can affect its output.



(a) ReLU Activation Function

(b) Hill Activation Function

Figure 8: Two examples of activation functions. The Rectified Linear Unit (ReLU) function (Equation 7) in Figure 8a is a popular activation function in computer science. Figure 8b depicts a typical Hill function (Equation 8), a popular equation to model biochemical reactions, such as gene expression. The relevant parameters are V_{max}, which is the maximum output potential, K_A, which is the required input to reach $V_{max}/2$, b_0, the background expression level, and n, the cooperativity. Figure 8b uses a K_A of 2, a V_{max} of 1, a b_0 of 0, and iterates through 5 different values of n.

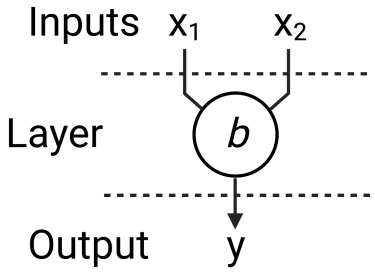
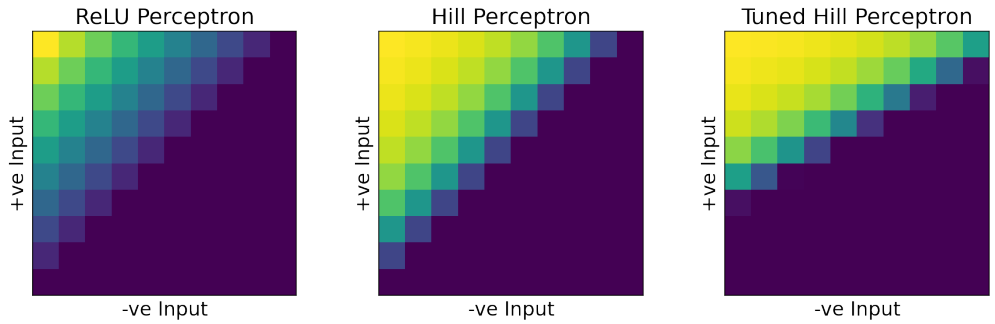


Figure 9: Simplified schematic of a two input, single layer perceptron with inputs x_1 and x_2, bias b and output y. This simple perceptron can be described by Equation 9, where one weight is positive (w_1), and one weight is negative (w_2).

$$y = f(b + w_1 x_1 - w_2 x_2) \qquad (9)$$



(a) Perceptron with ReLU activation function

(b) Perceptron with Hill activation function

(c) Perceptron with modified weights and bias

Figure 10: Example output of three differently-tuned perceptrons that all follow Equation 9. All perceptrons take two inputs, one of them being positive (y axis), the other negative (x axis). To illustrate possible outputs, the three figures differ in activation function, weights, and biases. Figure 10a uses the Rectified Linear Unit (ReLU) function (Equation 7), a popular function used for neural networks in computer science. The weights are both set to 1, and the bias is 0. ReLU does not require any other parameters. Figure 10b uses a Hill function (Equation 8) as the activation function, with a k_A and cooperativity (n) of 2, and a V_{max} of 1. Both weights are 1, and the bias is 0. Figure 10c shows how changing the weights can modify the output of a perceptron. The same parameters and activation function as in Figure 10b are used, but the weight associated with the positive bias is set to 1.8, and a bias of -5 is introduced.

1.3.3 Building a Molecular Neural Network

Over the past few years, molecular neural networks have been implemented in chemical systems using sequestration reactions and strand displacement reactions with DNA (Qian et al., 2011; Cherry and Qian, 2018; Okumura et al., 2022), or other cell-free systems (Pandi et al., 2019). The first engineered molecular neural network was built with DNA, and based on DNA strand displacement reactions. A single perceptron was split into three subfunctions, all based on seesaw gates built with toehold mediated strand displacement. The three connected seesaw gates separately dealt with the multiplication part ($w_i x_i$) and the summation part ($\sum_{i=0}^{m} w_i x_i$) of the perceptron. The weighted sum was then compared to an arbitrarily defined threshold th ($\sum_{i=0}^{m} w_i x_i \geqslant th$). Weights of the network were predefined by the output level of the multiplying gate, which was set by the ratio between the gate and output DNA abundances (Qian et al., 2011). More recently, researchers have been trying to build molecular neural networks in cells (Rizik et al., 2022; Racovita et al., 2022; Chen et al., 2022).

Rizik et al. (2022) built a molecular neural network in *E. coli*. Instead of using molecular sequestration reactions, regular gene activation and inhibition via transcription factors were used (Meyer et al., 2019). To get typical (linear) addition and subtraction behaviour from feedback topologies with exponential behaviour, all data was log transformed, yielding a two-

input power law and multiplication perceptron (Daniel et al., 2013; Rizik et al., 2022). The weights of the perceptron were determined by the network topology of the circuit, as well as the Hill coefficients of the transcription factor activation functions. The bias of the perceptron was determined by the maximum protein expression level (e.g. via promoter or RBS strength), and the intrinsic strength of the respective transcription factor. Closely matching artificial neural network architecture enabled weight optimisation via backpropagation and gradient descent (Rizik et al., 2022).

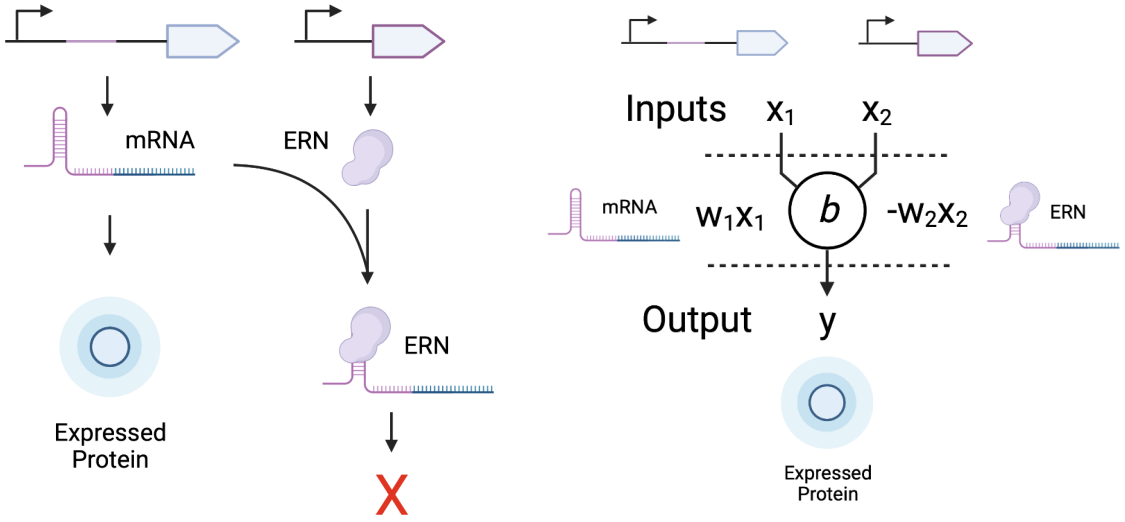
Another study demonstrated reinforcement learning without closely following a neural network structure, and distributing computing tasks between different colonies of *E. coli* to play tic-tac-toe (Racovita et al., 2022). By distributing computing, both biological burden and limitations because of a lack of biological parts were minimised, analogously to previous studies in Boolean networks (Tamsir et al., 2011; Al-Radhawi et al., 2020). Instead of implementing a network structure, the authors designed adaptable memory units, that could store an analogue weight based on an active/inactive plasmid ratio. The active and inactive plasmids had the same origin of replication, but had an active or inactive antibiotic resistance gene respectively. By punishing bad performers with the antibiotic, the ratio of the two plasmids could be shifted towards more of the active plasmid, which encoded a different fluorescent protein. Reading the induced fluorescent protein level gave a measure of the weight of the memory unit, which could be adjusted by reinforcement learning via antibiotics (Racovita et al., 2022).

Chen et al. (2022) built a winner-takes-all neural network in mammalian cells. The authors used *de novo* designed protein heterodimers to conditionally activate split viral proteases (Chen et al., 2019; Fink et al., 2019). The designed heterodimers enabled sequestration reactions that performed addition and subtraction. The input heterodimer had a very strong interaction with the correct pairing, and a strong interaction with an adapter heterodimer. The adapter heterodimer had a weaker interaction with its adapter counterpart. One half of the input heterodimer was fused to the first half of the adapter heterodimer and to a half of the split protease. The other half of the split protease was only fused to the second half of the adapter heterodimer. Without input, the protease was inactive, since the adapter was bound by half of the input heterodimer, and was not available to bind with the other half of the adapter protein. By adding the second half of the input dimer, the adapter protein was free to dimerise, and the protease became active. By using two sets of these protein fusions, the two-node network could compare which input was more abundant. By changing the ratios of the proteases, the weight of each input could be changed, and the decision boundary between

the two compared inputs could be tuned (Chen et al., 2022).

Although all these studies used different approaches, each had a way to store the weight of an interaction. Additionally, sticking to neural network structure enables more flexible computing, and enables computation in a single cell, which is critical for many applications, such as cancer classifiers or induction of morphogenic pathways (Abiodun et al., 2019; Shakiba et al., 2021; Guye et al., 2016; Teague et al., 2016; Xie et al., 2011). Moreover, sequestration reactions have successfully been used to create weighted addition and subtraction, both in living cells and in *in vitro* systems (Qian et al., 2011; Cherry and Qian, 2018; Okumura et al., 2022; Chen et al., 2022). While the presented studies already show progress towards molecular neural networks operating in cells, some foundational steps are yet to be taken. Firstly, no implementation has used a network structure including a hidden layer. This greatly limits classification potential, especially when only simple activation functions (such as ReLU) are used. Secondly, the presented studies either display neural network structure, or have inducibly changeable weights - no strategy to date has been developed, where the weights of a multi-layer network can be permanently changed in an inducible manner.

In this thesis, RNA binding endoribonucleases (ERNs) of the Cas6 and Cas13 families are used to cut mRNA. Since this reaction is mainly irreversible, one RNA molecule is being sequestered by one ERN, leading to a subtraction. The subtraction occurs, because the RNA gets inactivated by cleaving of the 5' cap off the RNA molecule by the ERN. The irreversible binding of the ERN to the RNA molecule prevents the ERN to cut several RNA molecules, which leads to the desired subtraction via RNA sequestration. Several of these ERNs have been characterised and published as part of the PERSIST toolbox developed by another PhD student in the Weiss lab (DiAndreth et al., 2022). Regulation at the RNA level has the advantage that constitutively active promoters such as CAG, hEF1α and CMV can be used. Using these constitutive promoters enables future implementations that are stably integrated in the genome, while ensuring stability of expression across multiple cell passages and cell lines with minimal gene silencing. Other promoters that are downregulated transcriptionally will lose the ability to turn on after an extended period of being inactive via gene silencing (Cabrera et al., 2022; DiAndreth et al., 2022). Weighted molecular addition is easily implemented via including more copies of a specific gene expression cassette. Barring resource competition, two copies of the same gene should produce twice the amount of RNA, and thus twice the amount of output protein. Including multiple copies of the same gene to tune a weight or an input has already been successfully implemented in the context of molecular machine learning algorithms discussed above (Racovita et al., 2022; Chen et al.,



(a) Overview of the PERSIST mechanism

(b) PERSIST in the context of a perceptron

Figure 11: Overview of the PERSIST mechanism. Figure 11a shows the sequestration reaction of an mRNA molecule. Binding of the ERN to its recognition sequence leads to cutting of the 5' cap of the mRNA, which in turn leads to degradation of the mRNA and no gene expression. Figure 11b shows this sequestration reaction in the context of a perceptron. Multiple copies of either the output gene or the ERN gene are the positively and negatively weighted inputs respectively.

2022). The weight of addition can be tuned by several well understood mechanisms, such as by varying promoter strength or RBS strength in bacterial systems, or by the inclusion of upstream open reading frames (uORFs) in mammalian systems (Salis, 2011; LaFleur et al., 2022; Ferreira et al., 2013). Figure 11 illustrates how ERNs of the PERSIST platform are used to sequester RNA molecules, and how this can be used to construct a molecular perceptron. The perceptron shown here shows the minimal working example of a single, two-input node, with one negative and one positive weight.

In summary, the below conditions must be fulfilled in order to design a molecular neural network:

1. Inputs must be able to be added and subtracted from each other via sequestration.
2. Addition and subtraction must be able to be weighted, and the weights should be tuneable.
3. Multiple nodes should be able to be constructed and follow the same architecture. The nodes should be able to be cross linked, and not display any unwanted cross-talk.

1.4 Aims

My main drive during this research was to gain a deeper appreciation for the complexity of biological decision making. The aim of this thesis is split into two main parts: Firstly, to

engineer and test a biological circuit that employs the Turing mechanism to generate gene expression patterns. Since no previous study has built deterministic Turing patterns from the ground up before, I hope to better understand how this mechanism is implicated in biological pattern formation. Additionally, this engineering effort could bring spatial patterning closer to application, since the circuit is implemented from first principles. Secondly, I aim to build neuromorphic devices that can function in the context of a molecular neural network, unlocking a scalable architecture for biological decision making. In this thesis, I hope to lay out a foundation for both pattern generation and pattern recognition. In the future, convergence of these principles could allow biological engineers to build cells that can decide where, when and how to express certain genes, enabling macro-scale engineering of biology. This main motivation is broken down into the following aims:

1. **Engineer synthetic Turing patterns.** As mentioned in Section 1.2, Turing patterns are a well defined mechanism that can generate a diverse number of different gene expression patterns, ranging from spots, through to labyrinths and stripes. To build Turing patterns, I propose these four sub-aims, where each marks a milestone towards building a reaction diffusion pattern generator.

    (a) *Make sure network connections are intact.* The first step is to determine whether the engineered circuit is functional. The circuit topology was chosen based on Scholes et al. (2019), which predicted that certain circuits have more parametric robustness over other topologies. A variation of the most robust circuit, 3954, is implemented with well characterised parts. I set out to confirm that the parts still function in this context.

    (b) *Test the diffusors.* The second step for engineering synthetic Turing patterns is to make sure that the chosen diffusors can act as morphogens in the chosen context. In this thesis, I aim to determine whether the diffusors act on a relevant timescale, so they can be used in a reaction-diffusion pattern. Additionally, an indication of the differential diffusivity between the two diffusors of the pattern should be established. Since Turing patterns typically require a difference in diffusion of the two molecules, this can inform whether the chosen molecules are adequate.

    (c) *Screen for spatial patterns.* While addressing the previous two sub-aims can give an indication whether pattern formation is possible, the only way to screen for gene expression patterns is to observe them directly. The assembled circuit is therefore screened via microscopy. Additionally, circuit variations are constructed and screened that should not produce classical Turing patterns.

(d) *Mathematical analysis.* Since the parametric robustness of Turing patterns is very low, related mathematical models are explored to explain patterning behaviour. A special focus is given to growth of the reaction space, which is often neglected in previous engineering studies.

2. **Build a molecular neural network.** Neural networks are the most popular algorithm for pattern recognition in computer science. Several neuromorphic components are developed in order to build a small molecular neural network. I identified the following sub-aims, where each brings molecular neural networks closer to maturity.

   (a) *Build molecular perceptrons.* Molecular sequestration reactions using the PERSIST mechanism are tested to determine their viability to construct molecular perceptrons. Three of these proteins are tested, so a small molecular neural network with three nodes can be built. Importantly, both negative and positive inputs should be possible.

   (b) *Design a library of genetic weights and biases.* One of the main advantages of neural networks is their ability to perform diverse classification tasks with the same network architecture, simply by tuning the weights of the connections between the nodes of the network. To enable this, three different strategies are explored to tune genetic weights. The genetic weights are then tested to tune a molecular perceptron. Further experiments are carried out to test molecular biases, which enable the same tuneability for molecular perceptrons as for their counterparts in computer science.

   (c) *Construct a multi-layer molecular perceptron.* In order to be a universal approximator, multiple perceptrons need to be strung together in a network. I aim to build two simple molecular neural networks that consist of multiple molecular perceptrons. The first network consists of two molecular perceptrons in a single layer to form a dual region classifier. The second network consists of three molecular perceptrons, two of which are operating in a hidden layer, resulting in a 2D bandpass filter.

   (d) *Update genetic weights.* As mentioned, the same neural network architecture can be used for diverse classification tasks, if the weights and biases are tuned. Using existing assembly strategies, permanently adjusting the weights and biases of a network would require the network to be built again from the ground up. I aim to establish an inducible strategy, where the weights of a molecular perceptron can be permanently adjusted without molecular cloning using prime editing.

2

Pattern Generation: Engineering Synthetic Turing Patterns

In this Chapter, I used a gene circuit (3954) based on the Turing mechanism to generate spatial gene expression patterns in *E. coli*. I first confirmed that the assembled circuit does not contradict the intended design in liquid culture, and then measured the effective diffusivity of the morphogens of the circuit. Furthermore, I screened the assembled circuit 3954, as well as control subcircuits, via microscopy. In an attempt to explain some unexpected patterns, I used mathematical modelling to investigate the effects of domain growth on pattern formation.

2.1 Circuit Design

As mentioned in Section 1.2.3, Turing patterns are predicted to be very sensitive to changes in parameters, and several strategies have been explored mathematically to minimise this problem. One important design decision which heavily impacts robustness is the choice of circuit topology. In this case, the circuit topology is based on a study from a previous PhD Student in the Isalan lab, which explored all networks with two diffusors and three nodes (Scholes et al., 2019). The circuit shown in Figure 12 - circuit 3954 - showed good parametric and topological robustness, and was implemented using well characterised parts for *E. coli* (Brödel et al., 2016; Meyer et al., 2019; Du et al., 2020). A high level abstraction of the circuit is shown in the top right corner. These interactions can be abstractly modelled by Equation 10. As shown, the interactions are modelled by Hill functions, as already described by Scholes et al. (2019). To visualise the circuit, Node B (producing HSL variant $3OHC_{14}HSL$) and node C (the non-diffusing node) are producing GFP and mCherry respectively.

$$\frac{\partial[A]}{\partial t} = D_A\nabla^2[A] + b_A + Vmax_A \frac{1}{\left(\frac{[B]}{kA_{BA}}\right)^{n_{BA}} + 1} - \mu_A[A] \tag{10a}$$

$$\frac{\partial[B]}{\partial t} = D_B\nabla^2[B] + b_B + Vmax_B \frac{\left(\frac{[A]}{kA_{AB}}\right)^{n_{AB}}}{\left(\frac{[A]}{kA_{AB}}\right)^{n_{AB}} + 1} \frac{1}{\left(\frac{[C]}{kA_{CB}}\right)^{n_{CB}} + 1} - \mu_B[B] \tag{10b}$$

$$\frac{\partial[C]}{\partial t} = b_C + Vmax_C \frac{1}{\left(\frac{[A]}{kA_{AC}}\right)^{n_{AC}} + 1} \frac{1}{\left(\frac{[B]}{kA_{BC}}\right)^{n_{BC}} + 1} \frac{\left(\frac{[C]}{kA_{CC}}\right)^{n_{CC}}}{\left(\frac{[C]}{kA_{CC}}\right)^{n_{CC}} + 1} - \mu_C[C] \tag{10c}$$

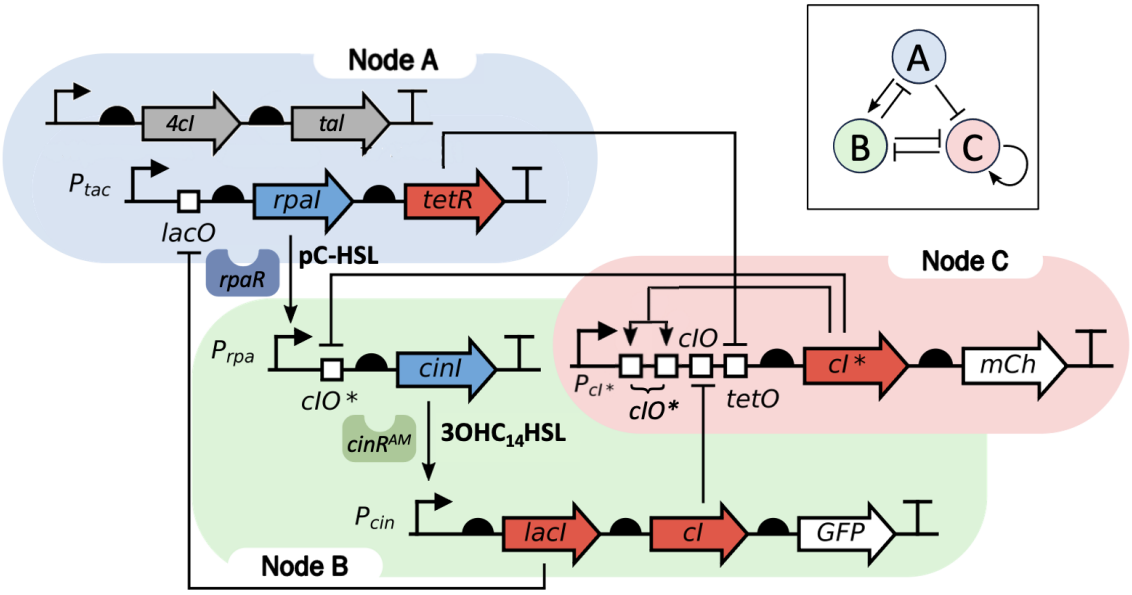


Figure 12: Schematic of the full engineered circuit. The circuit design is inspired by circuit 3954 as discovered via mathematical modelling by Scholes et al. (2019). A high-level circuit abstraction is shown in the top right. The circuit is implemented using previously well characterised parts. *lacI*, *tetR*, *cI* and *cinI* are well characterised in Meyer et al. (2019). *rpaI* is characterised by Du et al. (2020). An orthogonal λ bacteriophage *cI* variant, noted as cI^*, is characterised in Brödel et al. (2016). Enzymes producing small diffusive molecules are shown in blue, transcription factors in red, and fluorescent reporters in white. Interactions between active circuit components (blue and red) are indicated by arrows between components. The figure uses standard SBOL visual (Baig et al., 2020). The initial circuit was constructed by Jure Tica and Tong Zhu (2020, unpublished results). The abstracted circuit can be modelled with Equation 10.

Notably, the most robust circuit found in Scholes et al. (2019) also has a positive feedback on node A (producing N-(p-coumaroyl)-L-homoserine lactone (pC-HSL)); this feature is absent in the current implementation, due to constraints encountered during implementation (Tica and Zhu, 2020, unpublished results). Importantly, the circuit still has one non-diffusing node (node C), allowing for equal diffusivity of the activator (pC-HSL on node A), and the

inhibitor (3OHC$_{14}$HSL on node B) (Klika et al., 2012; Scholes et al., 2019). This is a critical feature, because the chosen diffusors are predicted to diffuse with similar diffusion rates in D_2O (Evans et al., 2018; Oliver Huidobro et al., 2021). The diffusion rates were calculated using a Stokes-Einstein Gierer-Wirtz estimation method, which was originally developed to predict diffusion constants in deuteriated solvents (such as D_2O) for diffusion-ordered NMR spectroscopy. Because the calculation requires an experimentally determined constant for each solvent, the diffusion coefficients could not be predicted in H_2O. There might be a slight discrepancy between these predictions in D_2O and H_2O, due to the stronger hydrogen bonding in D_2O, and its increased weight. Diffusion in H_2O should theoretically be marginally faster than predicted for D_2O. Equation 11 shows how the molecular weight (of small biological molecules weighing $\leqslant$1.5 kDa) is correlated with the diffusion coefficient. The terms in the equation are defined as follows: D: Diffusion coefficient. k_BT: Thermal driving force at temperature T in Kelvin. η: Viscosity. ρ_{eff}: Effective density. This parameter was fit to a training set. r and r_S: Effective solute and solvent radius respectively, where the radius is approximated for a sphere based on the molecular weight of the molecule. MW and MW_S: Molecular weight of the solute and solvent respectively (Evans et al., 2018). The parameter $\alpha$is the ratio of the approximated solvent and solute radius from their respective molecular weights. Generally, small molecules all diffuse slower in biofilms; this scaling may also not be linear, and molecules can diffuse up to ten times slower in biofilms than in water (Rani et al., 2005). Additionally, the effective diffusivity of signalling molecules in their biological context can differ greatly based on their downstream processing. For instance, signalling molecules that enter the cell irreversibly when they activate gene expression have a shorter range than molecules that can be reused (van Gestel et al., 2021). Nevertheless, this calculation can provide a naive starting point to estimate a diffusion coefficient if measurements in a biological context are unavailable. Importantly, the two chosen diffusors have been demonstrated to display minimal crosstalk (PhD thesis by Jure Tica, unpublished results).

$$D = \frac{k_BT\left(\frac{3\alpha}{2} + \frac{1}{1+\alpha}\right)}{6\pi\eta\sqrt[3]{\frac{3MW}{4\pi\rho_{\text{eff}}N_A}}} \tag{11a}$$

$$\alpha = \frac{r_s}{r} = \sqrt[3]{\frac{MW_S}{MW}} \tag{11b}$$

Apart from the two enzymes producing the small diffusing molecules (*rpaI* producing pC-HSL and *cinI* producing 3-hydroxytetradecanoyl-homoserine lactone (3OHC$_{14}$HSL) on node A and node B respectively), a low copy (pCC1 origin) plasmid constitutively expresses the

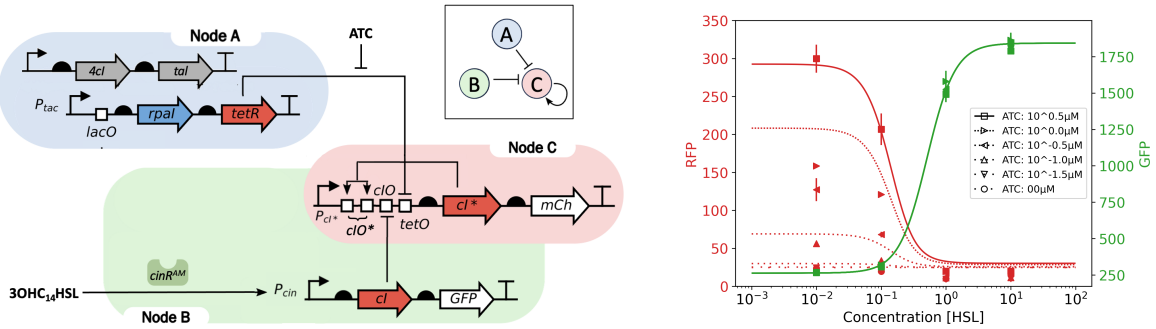
sensors for pC-HSL (*rpaR*) and 3OHC$_{14}$HSL. The supporting pCC1 plasmid encoding the required receptors is shown in its entirety in Section 5.21. Notably, the regulated *rpaI* enzyme on node A only synthesises the first step during pC-HSL production. The other two required enzymes (*4cI* and *tal*) are expressed constitutively on the node A plasmid. The circuit also encodes four well characterised transcription factors: *tetR* on node A, *lacI* and *cI* on node B, and *cI5G6G,P* (here labelled cI^*) on node C. *tetR* acts as a repressor from node A to node C by binding the tet operator *tetO*. The *tetR* repression can be inhibited by adding anhydrotetracycline (ATC), a derivative of the antibiotic tetracycline without antibiotic activity. *lacI* is an inhibitor acting from node B to node A, and is inhibited by isopropyl β-D-1-thiogalactopyranoside (IPTG) (Meyer et al., 2019). *cI* is expressed from node B and inhibits node C. A previous study used a directed evolution approach to find an orthogonal *cI* variant, *cI5G6G,P* (Brödel et al., 2016). In this thesis, the *cI5G6G,P* variant is labelled cI^* for readability. cI^* is expressed on node C, and activates node C, as well as inhibits the production of *cinI* on node A. The repression and activation on node C is due to three separate cI operators. The first two (reading 5'→3') bind cI^*, and facilitate RNA polymerase binding to activate transcription. The third operator binds the wild type *cI* from node B, which stops gene expression by blocking RNA polymerase binding. Apart from transcription factors, node B and node C are labelled with *GFP* and *mCherry* respectively for visualisation. Apart from the genes expressed in the circuit, several enzymes are expressed constitutively, such as transcription factors for sensing the produced small molecules pC and 3OHC$_{14}$HSL, and degradation enzymes for LVA and ASV degradation tags (Meyer et al., 2019). Each part used in this circuit is described in detail in Section 5.19, and a source for each part is provided. The molecular structure both HSL molecules is also shown in Section 5.19. In total, the entire circuit consists of four plasmids; one medium copy number (20-40 copies per cell) plasmid per node (pCOLA for node A, pCDF for node B, and pET for note C), and one low copy plasmid (bacterial artificial chromosome (BAC) vector pCC1) for the constitutively expressed receptors for the diffusors (Milo et al., 2010). For a complete annotated plasmid list, see Section 5.21.

2.2 Subcircuit Characterisation

The circuit shown in Section 2.1 is composed entirely of previously well characterised parts (Brödel et al., 2016; Meyer et al., 2019; Du et al., 2020; Oliver Huidobro et al., 2021). However, two controls were built to determine whether the circuit is in a functional range when in context of the circuit described in Section 2.1. Both controls were characterised in liquid culture, as described in Section 5.16. Additionally, both controls were designed

to have coherent logic - following the logic of an input, the circuit should only have one effect, and not split into multiple possible outcomes. The controls were not designed to yield accurate parametrisation of circuit 3954 as shown in Equation 10, but rather to confirm the connections between nodes. This approach was taken because of several reasons: Firstly, the context between growing colonies (the setup used to screen for pattern formation) and liquid culture is widely different. Since Turing patterns are predicted to be very parameter sensitive, any parameters obtained from liquid culture may not inform whether Turing patterns are possible or not. Secondly, while parametrisation may not be informative, this approach can identify broken connections between nodes, and highlight whether the circuit topology can be captured by the model shown in Equation 10. Models for the subcircuits were based on the full circuit, and were fit to the data in order to highlight discrepancies between the model and the data. This can inform whether the model for circuit 3954 (Equation 10) must be rejected. Lastly, the participating transcription factors and enzymes cannot be measured directly, but were measured via proxy fluorescent proteins. While a linear correlation between proteins from the same transcription cassette can be assumed, not all transcription cassettes have a directly responding fluorescent protein. Additionally, an enzyme is present that produces a participating molecule, further limiting accurate parametrisation.

The first control circuit is shown in Figure 13a. The control circuit closely mimics the full circuit (3954) shown in Figure 12. However, the gene expression cassette encoding *cinI*, the enzyme that produces $3OHC_{14}HSL$, was removed. To prevent incoherence in the circuit, *lacI* was also removed from the circuit. ($3OHC_{14}HSL \rightarrow lacI \dashv tetR \dashv$ node C and $3OHC_{14}HSL \rightarrow cI \dashv$ node C.) In this control, one of the two diffusors in the circuit ($3OHC_{14}HSL$) was applied exogenously, activating the remaining node B, which should lead to an increase in node B. Additionally, different ATC concentrations were sampled. ATC represses the function of *tetR*, which is constitutively expressed, thereby activating node C. Node B is unaffected by ATC, since node C does not feed into node B in this circuit variation. The expected output for this circuit is an increase of GFP as more $3OHC_{14}HSL$ is added. As GFP increases, RFP output should decrease, since node B represses node C. However, this increase in RFP should only be visible at high ATC concentrations, since a constitutively active node A should be able to shut off node C if left unchecked. Data collected in liquid culture (Figure 13b) confirmed that the tested part of the circuit functions as expected.



(a) Circuit Schematic

(b) Dose Response

Figure 13: Characterisation of a subcircuit of circuit 3954 (Figure 12). A schematic representation of the characterised subcircuit is shown in Figure 13a. In contrast to circuit 3954, node B does not contain *cinI*, the enzyme producing 3OHC$_{14}$HSL, and *lacI*. The subcircuit is characterised with two inputs: 3OHC$_{14}$HSL and ATC. Figure 13b shows the dose response of the subcircuit to 3OHC$_{14}$HSL and ATC. 3OHC$_{14}$HSL is plotted on the x-axis versus green and red fluorescent output divided by OD600 in arbitrary units on the green and red y-axes respectively. Different ATC concentrations are shown as differently shaped markers. Errorbars show the standard deviation around the mean of two technical replicates. The results from the best fit of Equation 12 are shown as lines, where dots are spotted more sparsely as the ATC concentration decreases. Parameters for the model, as well as more data sets can be found in Appendix A.

$$0 = b_B + Vmax_B \frac{\left(\frac{[HSL]}{kA_B}\right)^{n_B}}{1 + \left(\frac{[HSL]}{kA_B}\right)^{n_B}} - B^* \tag{12a}$$

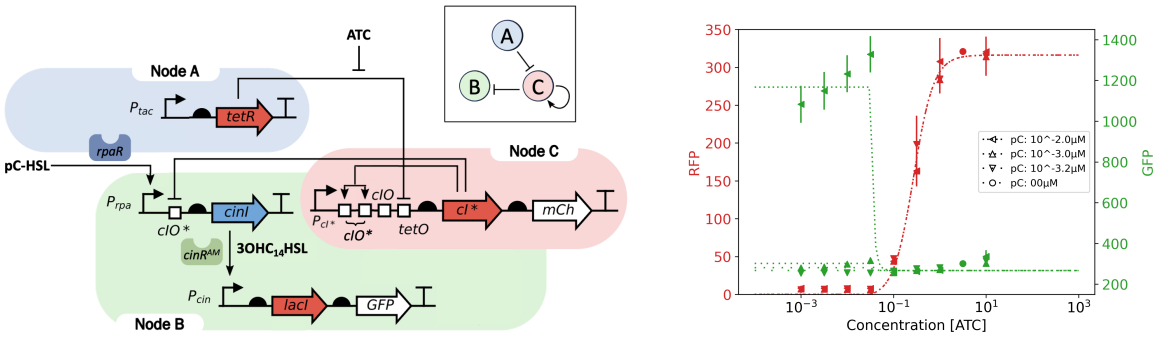
$$0 = b_C + Vmax_C \frac{1}{1 + \left(\frac{[B*]}{kA_{BC}}\right)^{n_{BC}}} \frac{\left(\frac{[ATC]}{kA_C}\right)^{n_C}}{1 + \left(\frac{[ATC]}{kA_C}\right)^{n_C}} \frac{\left(\frac{[C*]}{kA_{CC}}\right)^{n_{CC}}}{1 + \left(\frac{[C*]}{kA_{CC}}\right)^{n_{CC}}} - C^* \tag{12b}$$

To further investigate whether the data contradict any assumptions made during modelling, the data were fit to the model equations at steady state shown in Equation 12. The lines connecting the markers in Figure 13b show the best fit for the system of equations. Note that all points were fit simultaneously for the system of equations - i.e. the parameters for the GFP line were used to calculate the steady state for node B, which was then fed into the steady state for node C. All different ATC concentrations also share their parameter set, and only differ in ATC concentration. Since a fit between the markers with the equations shown in Equation 12 was possible, the assumptions made for the model were not rejected. However, the quality of the fit in the red channel suffered at low 3OHC$_{14}$HSL concentrations. This highlights another issue with this type of characterisation: Even with a minuscule increase of GFP, RFP already becan to drop. This indicates that *cI* was already increasing, even

though the correlated GFP was not measurable, suggesting GFP has a detection limit that is too high to allow accurate characterisation and parametrisation of the circuit. This high detection limit is likely due to the high background of the media in the green channel, and the sensitivity of the plate reader. Biological repeats of the circuit are shown in Appendix A.

The second control circuit is depicted in Figure 14a. The circuit again closely mimics the full circuit shown in Figure 12. The control differs from the full circuit in two main ways: Firstly, the enzyme that produces pC-HSL, *rapI*, is absent from the control. Secondly, incoherence in the circuit was eliminated by removing the *lacO* site that enables the connection from node B to node A. The eliminated incoherence is due to an incoherent feedback loop: pC-HSL $\rightarrow$ 3OHC$_{14}$HSL $\rightarrow$ *lacI* $\dashv$ node A $\dashv$ node C $\dashv$ 3OHC$_{14}$HSL, which can be condensed into pC-HSL $\rightarrow$ 3OHC$_{14}$HSL $\dashv$ 3OHC$_{14}$HSL.

Additionally, a positive feedback loop was eliminated to further facilitate characterisation by deleting *cI*, which enables repression from node B to node C. The eliminated positive feedback is due to a double negative motif between node B and node C: pC-HSL $\rightarrow$ 3OHC$_{14}$HSL $\rightarrow$ *cI* $\dashv$ node C $\dashv$ 3OHC$_{14}$HSL. The circuit was again characterised with two inputs: ATC and pC-HSL; the dose response of the control in liquid culture is shown in Figure 14b. Note that, since pC-HSL has no predicted downstream effect on RFP output in this circuit, the ATC dose response is displayed on the x-axis for clarity. The data were again fit to a model that was based on the model of the full circuit. If the data are unable to reject the assumptions, they should fit to the system of equations shown in Equation 13. As with the other control circuit, all data was fit simultaneously. Notably, the GFP output (node B) is extremely sensitive to both ATC and pC-HSL input. This may be explained by the catalytic function of *cinI*. Since both node C, as well as pC-HSL act on the production of *cinI*, which produces multiple molecules of 3OHC$_{14}$HSL, the response is very steep. This behaviour was also captured in the model. Biological repeats with more granularity of pC-HSL input are shown in Appendix A. Although the model seems to capture the behaviour of the circuit, the slight increase of GFP before it sharply drops could not be explained by this simple mechanistic model. This behaviour was consistent between biological repeats (shown in Appendix A).



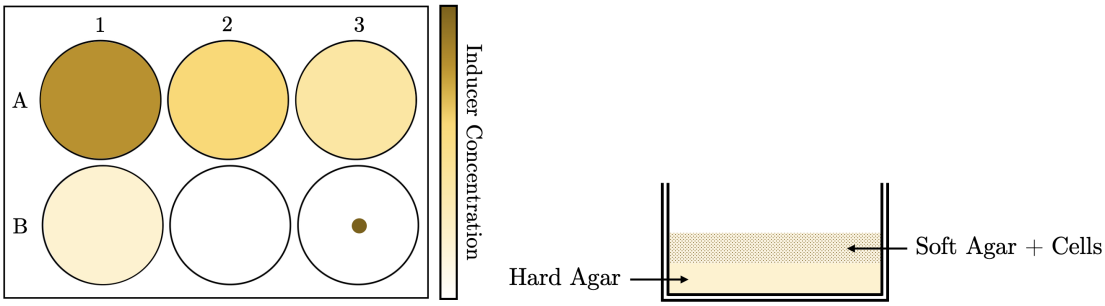
(a) Circuit Schematic (b) Dose Response

Figure 14: Characterisation of a subcircuit of circuit 3954 (Figure 12). A schematic representation of the characterised subcircuit is shown in Figure 14a. In contrast to circuit 3954, node B does not contain *cI*, and node A does not contain *rpaI*, the enzyme producing pC-HSL, nor the *lacO* operator. The subcircuit is characterised with two inputs: pC-HSL and ATC. Figure 14b shows the dose response of the subcircuit to ATC and pC-HSL. ATC is plotted on the x-axis versus green and red fluorescent output divided by OD600 in arbitrary units on the green and red y-axes respectively. Different pC-HSL concentrations are shown as differently shaped markers. Errorbars show the standard deviation around the mean of two technical replicates. The results from the best fit of Equation 13 are shown as lines, where dots are spotted more sparsely as the pC-HSL concentration decreases. Parameters for the model, as well as more data sets can be found in Appendix A.

$$0 = b_C + Vmax_C \frac{\left(\frac{[ATC]}{kA_C}\right)^{n_C}}{1+\left(\frac{[ATC]}{kA_C}\right)^{n_C}} \frac{\left(\frac{[C*]}{kA_{CC}}\right)^{n_{CC}}}{1+\left(\frac{[C*]}{kA_{CC}}\right)^{n_{CC}}} - C* \tag{13a}$$

$$0 = b_B + Vmax_B \frac{\left(\frac{[pC]}{kA_B}\right)^{n_B}}{1+\left(\frac{[pC]}{kA_B}\right)^{n_B}} \frac{1}{1+\left(\frac{[C*]}{kA_{CB}}\right)^{n_{CB}}} - B* \tag{13b}$$

Overall, the circuit connections appear to be in tact, and in a biologically tuneable range. Reliable parametrisation of the circuit was however not possible. The main reason for this, is that only a slight change in measured green fluorescence already had a large effect in the measured red fluorescence. This complicated model fitting, since correlated enzymes and transcription factors already had a measurable downstream effect, while the fluorescence level indicated that they are not present. This is most apparent in Figure 13. Further complications include that the circuit is extremely sensitive to the addition of pC-HSL, since the enzyme *cinI* non-linearly amplifies its signal. However, since the circuit connections qualitatively functioned as expected, the circuit was further investigated for pattern formation.



(a) Plate Setup (b) Well Setup

Figure 15: Schematic of the assay setup. A base of 1 ml of hard 1.4% agar is poured in a standard size 6 well plate. Then 1 ml of a softer 0.7% agar suspension of *E. coli* cells are distributed on top. 5 wells in the setup contain fixed and homogeneous inducer concentrations (one of which being 0). A droplet containing known high concentrations of pC-HSL and $3OHC_{14}HSL$ is then spotted in the middle of a well containing otherwise no inducer. Fluorescent output is then measured over time.

2.3 Characterisation of Diffusive Components

Apart from determining whether the connections within the circuit function as expected (Section 2.2), Turing pattern formation relies on the diffusivity of the activator and the inhibitor. To determine the effective diffusion coefficients of pC-HSL and $3OHC_{14}HSL$ in agar, the following setup was used: A base of 1 ml of hard 1.4% agar was poured in a standard size 6 well plate. Then 1 ml of a softer 0.7% agar suspension of *E. coli* cells was distributed on top (see Figure 15). The cells contained transcription factors that respond to pC-HSL and $3OHC_{14}HSL$, and produce mCherry and GFP respectively when binding their relevant promoters. An SBOL representation of the participating gene cassettes is shown in Figure 16. All relevant fluorescent data were collected using a TECAN plate reader as described in Section 5.17.

$$P^*(x) = P_{min} + P_{max}\frac{(\frac{x}{K_A})^n}{1 + (\frac{x}{K_A})^n} \tag{14}$$

5 wells in the setup contained fixed and homogeneous inducer concentrations. These were used to set up a standard curve and parametrise a Hill function (Equation 14). The parametrised function was then used to calculate an unknown inducer concentration from a fluorescent value. The results for the homogeneous wells, as well as the parameters for the best fit Hill function, are shown in Figure 17. The final well in the setup contained no homogeneous inducer. A droplet containing known high concentrations of pC-HSL and $3OHC_{14}HSL$ was then spotted in the middle of the well. The droplet spread out over time, and led to the production of fluorescent reporters (mCherry and GFP for pC-HSL and $3OHC_{14}HSL$

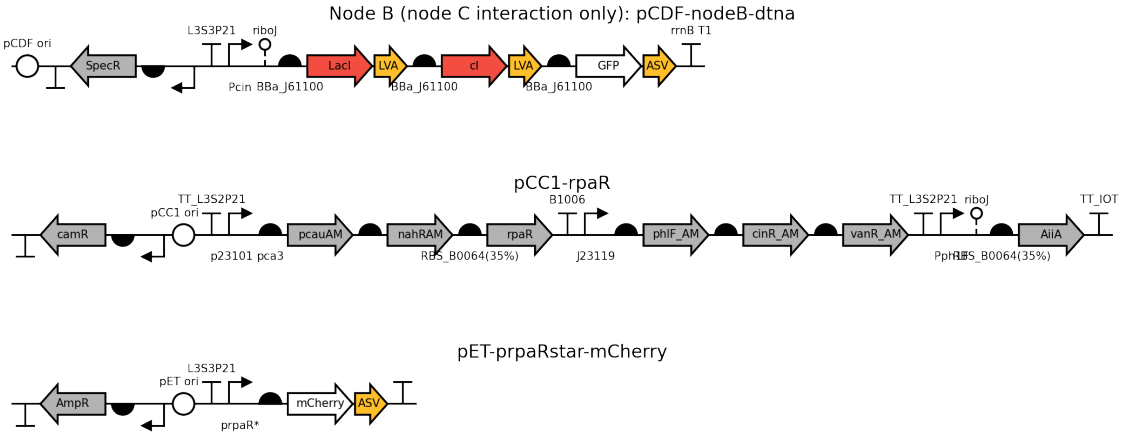


Figure 16: Plasmids used in the assays to characterise diffusion. The incomplete node B plasmid (Node B (node C interaction only): pCDF-nodeB-dtna) responds to 3OHC$_{14}$HSL and expresses GFP. pET-prapRstar-mCherry responds to pC-HSL and expresses mChery. pCC1-rpaR contains the transcription factors needed to sense 3OHC$_{14}$HSL and pC-HSL. The plasmids are used in the experiments that result in Figures 17, 18 and 19.

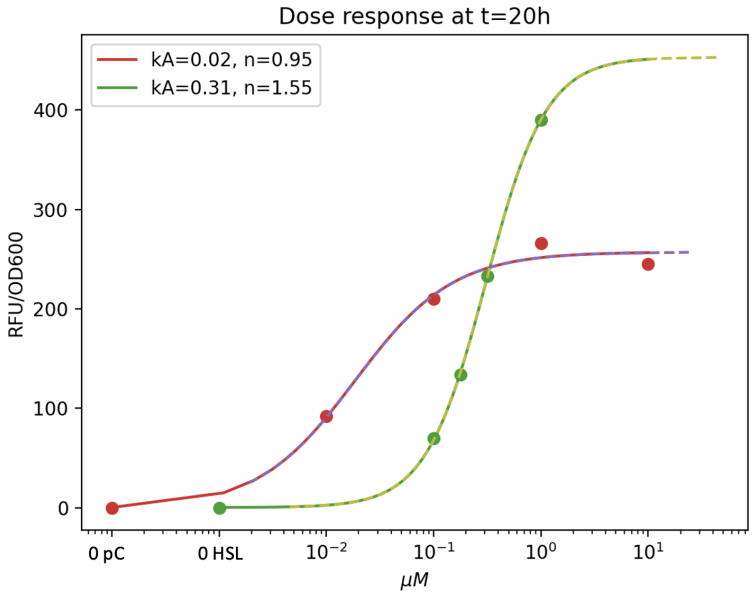


Figure 17: Dose response of 3OHC$_{14}$HSL and pC induction of GFP and mCherry respectively. A Hill function at steady state (Equation 14) was fit to the data points by least squares optimisation.

respectively). Spatially resolved fluorescent data was collected for the well with the spreading droplet. Using the parametrised Hill functions for mCherry and GFP fluorescence, the concentrations of pC-HSL and 3OHC$_{14}$HSL were calculated respectively. This resulted in a 2D diffusion profile, as shown in Figure 18. A 2D diffusion profile was also modelled numerically in the same simulated space, using Equation 15 (Fick, 1855). Both 2D profiles were then converted to 1D for easier fitting and visualisation. The x axis shows the distance in mm from the centre of mass of the diffusion profile. This also minimises inaccuracies that occurred when the spot of high concentration inducer was not placed exactly in the centre of the well. The y axis shows the concentration of inducer in µM. Both the diffusion coefficient D_C as well as the degradation constant μ_C were varied to fit Equation 15. This process yielded the diffusion coefficients $D_{pC} = 0.014mm^2h^{-1}$ and $D_{3OHC_{12}} = 0.0067mm^2h^{-1}$ for pC-HSL and 3OHC$_{14}$HSL respectively. This is more than an order of magnitude slower than the predicted diffusion coefficients for those molecules in D_2O, which are $D_{pC} = 2.47mm^2h^{-1}$ and $D_{3OHC_{12}} = 2.17mm^2h^{-1}$ (Evans et al., 2018; Oliver Huidobro et al., 2021). This large discrepancy may be explained by several factors.

$$\frac{\partial C}{\partial t} = D_C \nabla^2 C - \mu_C \tag{15}$$

One possible explanation is that diffusion in biofilms is already slower than in water. When fluorescent molecules with comparable sizes were examined in water versus in biofilms, the measured diffusion coefficient was up to ten times lower in biofilms (Rani et al., 2005). Since the measured diffusion was more than an order of magnitude slower than the predicted diffusion coefficients, this alone cannot account for this large discrepancy. An additional explanation is offered by the mechanism of action of these molecules. Quorum sensing molecules enter the cells, and then bind their receptor. The receptor thereby (temporarily) sequesters the molecule, effectively slowing down its diffusion. Additionally, the degradation gene for both HSL molecules (*AiiA*) is also expressed in the cells. This will effectively lead to irreversible uptake for at least some molecules, which has been linked to shorter range communication - i.e. slower diffusion constants (van Gestel et al., 2021). Since the aim of this assay was to determine the effective biological diffusion constants for these molecules - especially what their differential diffusivity is - it is not necessary to separate these two effects from the actual diffusion constant without cells present.

An additional factor that could influence the discrepancy between the measured and predicted diffusion coefficients could be an inherent inaccuracy in the assay. Close inspection of the biological data in Figure 19 reveals several other factors that may have influenced

this large discrepancy between measured and predicted diffusion coefficients. Firstly, the assay assumed that gene expression is a lot faster than diffusion, i.e. that a cell immediately responds (and adjusts) to a particular inducer concentration. Under this assumption, the fluorescent values were taken as steady state values, and could be mapped back from the steady state Hill function to chemical concentrations. This assumption might not hold true, especially at the edges of the diffusion profile, where cells were only recently exposed to inducer molecule. The dynamics of mCherry expression via pC-HSL induction occur over a timescale of several hours, which might be too slow to accurately represent the diffusion profile. While this concern needs to be kept in mind, the risk of this majorly impacting the results is still minimal. This is because the scale of the fastest measured effective diffusion coefficient ($D_{pC} = 0.014mm^2h^{-1}$) is still more than two orders of magnitude slower than the 10 hours it took for the *mCherry* fluorescence to reach its maximal value. Additionally, some of the differential diffusivity between pC-HSL and $3OHC_{14}HSL$ could be due to their difference in gene expression dynamics. This is not inherently problematic, since the effective diffusion between the two molecules in the context of a gene circuit also relies on how fast the circuit responds to its morphogens. However, while the assay might produce indicators of how much effective differential diffusivity is acting on the circuit, it may not be sufficient to determine the actual diffusion coefficients of the HSL molecules. Another concern that becomes apparent in the data shown in Figure 19, is the change in gene expression dynamics that occurred around 10 hours, where OD600 begins to decline. This shows that cell viability after 10 hours was impeded, and longer timescales could be unfavourable for gene expression. However, morphogenesis experiments, such as the ones shown in Section 2.4, were carried out over several days. Therefore, the timescale for this experiment was chosen to include this decline in cell viability, to better match the other experimental setups.

Overall, two main conclusions can be drawn from this diffusion assay. Firstly, while the predicted diffusion coefficients were both within 20% from each other, this was not the case for the measured effective diffusion, where the faster molecule diffuses more than twice as fast as the slower diffusor ($D_{pC} = 0.014mm^2h^{-1}$ and $D_{3OHC_{12}} = 0.0067mm^2h^{-1}$). Since higher differential diffusivity between the two morphogens is linked to higher probability of generating Turing patterns, this was a fortunate deviation from the prediction (Gierer and Meinhardt, 1972; Nakamasu et al., 2009). Secondly, the dynamics for gene expression are compatible with measuring and responding to a diffusion profile with these chemicals. If the diffusion rates were too fast, then the whole experiment would have just yielded a homogeneous concentration at the final timepoint. This would prohibit pattern formation,

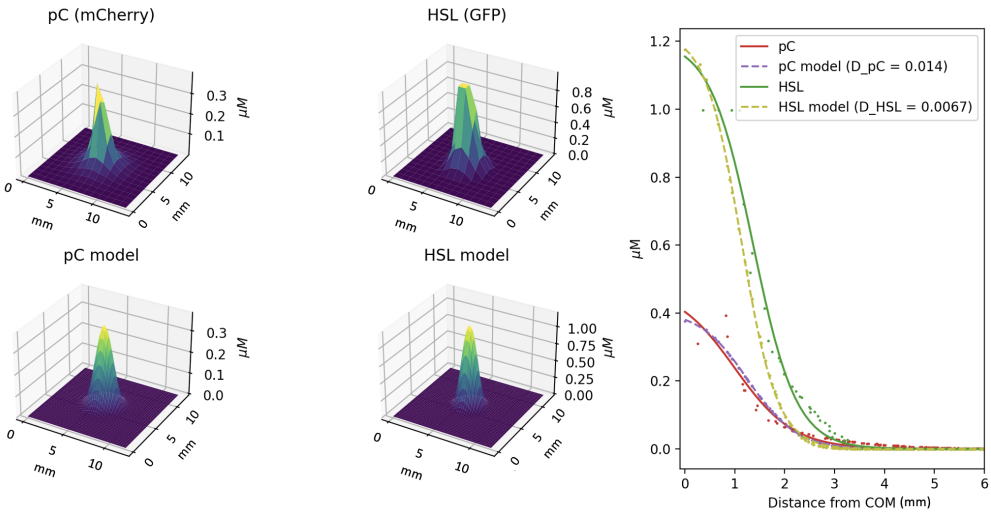


Figure 18: Results showing how a suspension of cells respond to a speading droplet of diffusor. A spatially resolved measurement of mCherry (for pC-HSL) and GFP (for $3OHC_{14}HSL$) is shown in the top row. The bottom row shows a 2D models of diffusing small molecules. The four 2D plots are converted back into µM from arbitrary fluorescent units using the standard curve shown in Figure 17. The 2D results are then collapsed to a 1D curve shown on the right by measuring the distance from the centre of mass (COM) on the x-axis, and concentration of the diffusor in µM on the y-axis. The pC and HSL diffusion models are then fit to the COM distance, resulting in diffusion constants $D_{pC} = 0.014mm^2h^{-1}$ and $D_{HSL} = 0.0067mm^2h^{-1}$. These results were obtained after 20 hours of incubation. The dynamics of the homogeneous expression wells are shown in Figure 19.

since the circuit needs time to respond to a difference between the diffusors.

2.4 Microscopy

2.4.1 Full Circuit

Since Turing instability is a binary phenomenon (either a pattern forms or it does not), the only way to know whether the engineered circuit is able to produce a pattern, is to image an area containing the engineered cells. For this purpose, single growing *E. coli* colonies containing the full circuit shown in Figure 12 were initially screened with a confocal microscope. This resulted in the pattern shown in Figure 20 (Zhu, 2020, unpublished results). The pattern displayed concentric rings forming around the centre of the colony, with thin green stripes antiphase to an otherwise predominantly red colony. When some of the cells were restreaked on another plate as a technical repeat, the stripes started forming again. Notably, the colony also had wedges without red, where the green stripes were still appearing. However, the concentric ring morphology was not readily reproducible with freshly transformed cells. After several attempts (>50) in the same experimental conditions (10 µM ATC, 1.4% 2xYTσ media in a 6 well plate), no repeated stripe pattern was found. The three main

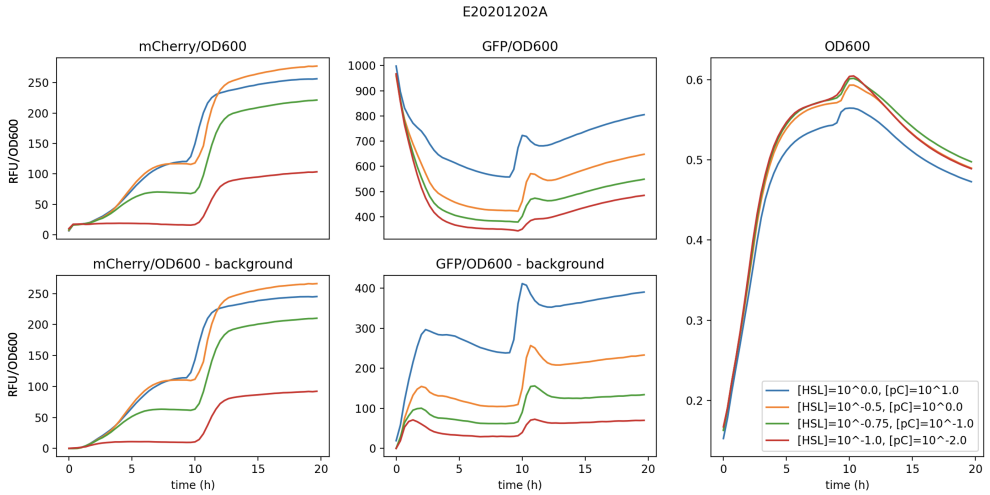
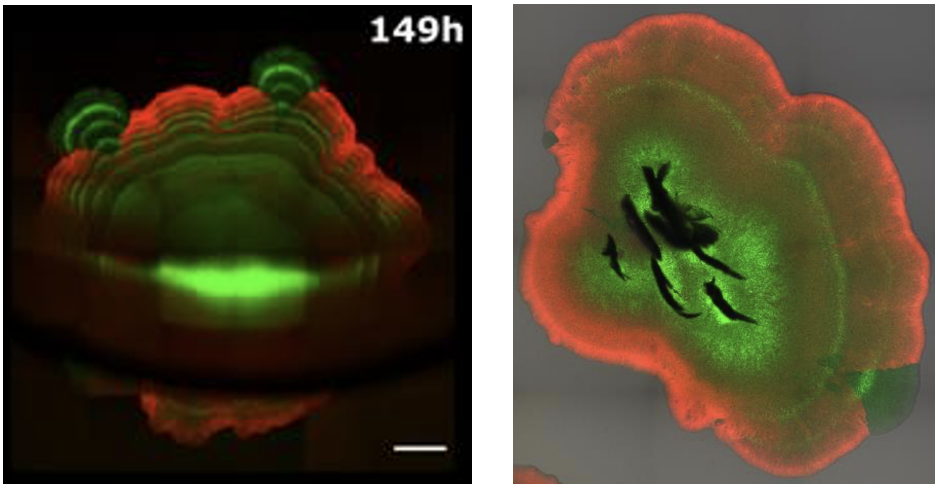


Figure 19: Time series results of the homogeneously induced wells used to calculate the standard curve shown in Figure 17. The right plot shows the OD600 measurement over time. The top of the smaller plots shows mCherry and GFP response over time in arbitrary fluorescent units, normalised by OD600. The bottom row subtracts the negative control from those lines to eliminate unwanted background signal. The figure legend is consistent between all five plots.

human controllable variables in the original assay were media type, ATC concentration and agar thickness, and were extensively screened to find a parameter regime that displayed repeatable patterning. To increase throughput, the screen was carried out in a 12 well plate, and colonies were spotted in the middle of the wells to isolate them.

One of the screened conditions in richer media (0.75 ml, 1.4% 3xYT agar) pushed the cells completely outside any patterning regime. This allowed for screening of the effects of ATC on the full circuit (shown in Figure 12). As seen in Figure 21, increasing ATC concentration led to a shift from predominantly green colonies to predominantly red colonies. This is to be expected, since ATC inhibits *tetR*, which inhibits node C activity, which is labelled with mCherry.

Screens across different media, agar thickness and ATC concentrations revealed a patterning regime with antiphase green spots, as shown in Figure 22. Interestingly, the spot size and frequency appears to be positively correlated with ATC concentration. The pattern was still sensitive to uncontrollable biological variables, and did not appear in every single screened colony. However, unlike the regular stripe pattern shown in Figure 20, the spots were reproducible from freshly transformed cells. Two controls that remove a diffusor and several interactions from the circuit were performed, to screen whether the spots appeared due to a reaction diffusion mechanism, or because of some other biological phenomenon.



(a) Original result (b) Restreak from colony

Figure 20: Initial confocal microscopy image of the full circuit (circuit 3954) as shown in Figure 12 (Zhu 2020, unpublished results). Figure 20a shows the first obtained results. Figure 20b shows a technical repeat of the experiment, where some of the cells from the colony in Figure 20a are restreaked on a fresh plate. Scale bar: 1 mm.

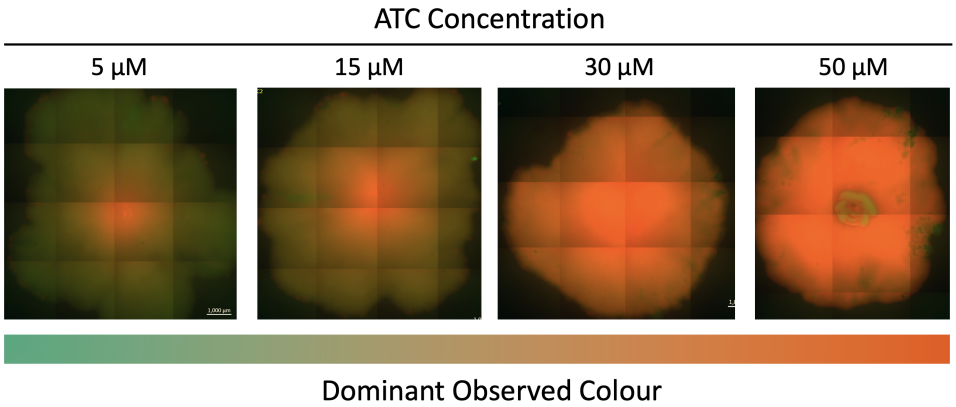


Figure 21: Microscopy results showing the behaviour of the full circuit when outside of a patterning regime. The images were taken on a wide field microscope approximately 4 days after seeding. Agar thickness and media type are consistent between all images (0.75 ml in a 12 well plate, 1.4% 3xYT agar). The assay was carried out according to the protocol described in Section 5.15.

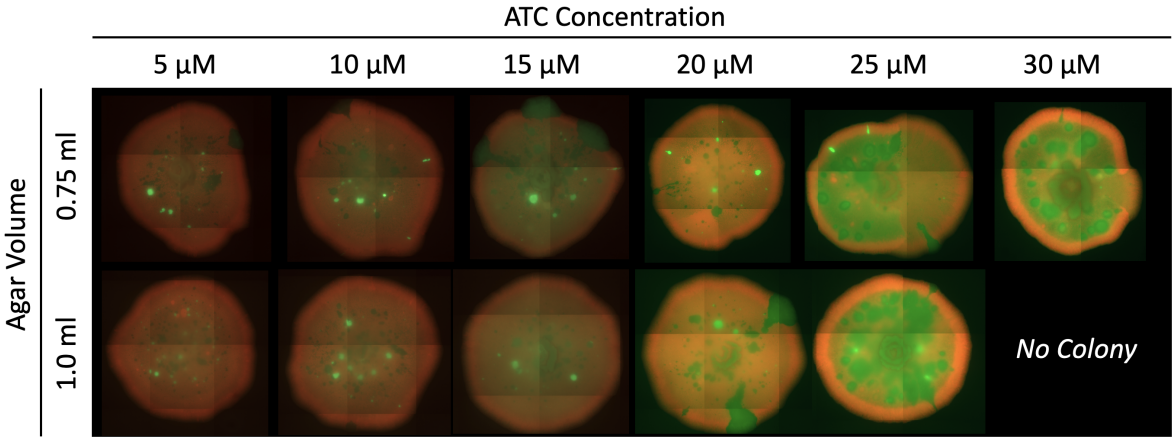


Figure 22: Results of microscopy screen across ATC concentrations. Green antiphase spots are increasing in size with increasing ATC concentration. The images were taken approximately 4 days after seeding on a wide field microscope. The assay was carried out in a 12 well tissue plate with 1.5% 2xYT agar, according to the protocol in Section 5.15.

2.4.2 Control 1: Removing Diffusor Interactions

In the first control, I removed the $3OHC_{14}HSL$ producing enzyme *cinI* from node B of the final circuit. Removing this gene expression cassette also removes the negative feedback from node C to node B, since the cIO^* operator acts upstream of the *cinI* gene. Additionally, the pC-HSL sensing promoter was also removed from node B. Therefore, by only removing a single gene, no diffusor was actively connected to the circuit anymore. This means any pattern observed with this circuit variant cannot be a reaction diffusion pattern, but can only be due to other effects, such as bistability.

This circuit variant was screened in different media conditions and agar thicknesses. A compilation of colonies that display a common morphology is shown in Figure 24. As can be seen, most colonies followed the same morphology, where a bright red ring formed on the outside of the colony, surrounding a centre of generally lower gene expression, which contained both mCherry and GFP. While this was the most common morphology, some biological repeats frequently formed green wedges in this condition. This may be due to bistability, where one cell switched from a predominantly red state to a predominantly green state. Since the state is preserved as the cell divides, the cells push outward, and produce a green wedge. One observed colony also produced "holes" of gene expression in the red channel. It is important to distinguish these holes from the antiphase spots that appear in the full circuit, as shown in Figure 22. Firstly, unlike the antiphase patterns of the full circuit, the holes had no effect on the fluorescence intensity of the green channel. Additionally, the colony that produced the holes in the red channel appears less healthy than other colonies

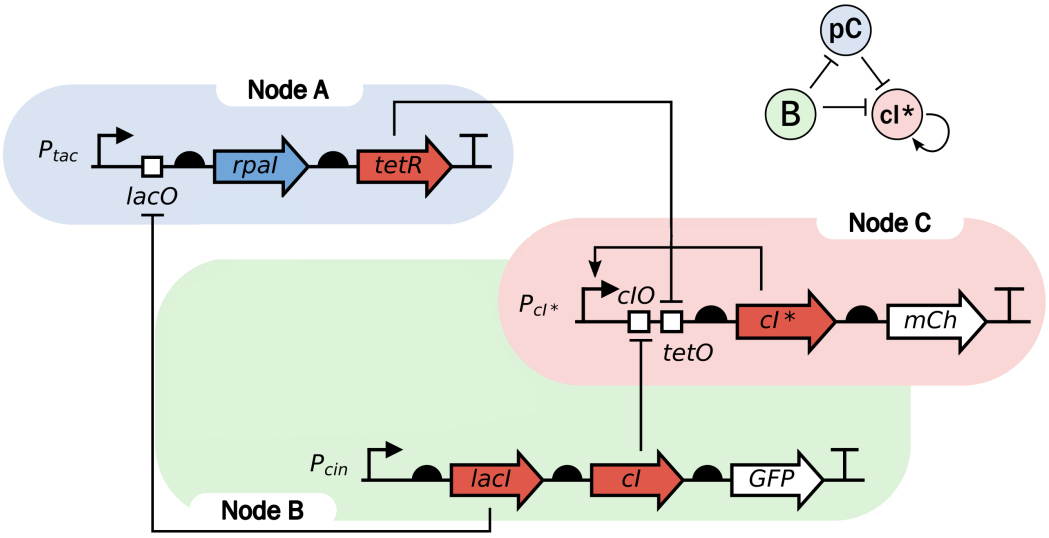


Figure 23: Circuit variation (control 1), which removes all diffusor interactions. The circuit differs from the full ciruit 3954 by removing the $3OHC_{14}HSL$ production enzyme *cinI* from node B. The pC-HSL sensing promoter is also absent; thus no diffusor has an effect on the circuit. A schematic representation of the circuit variation is shown in the top right. Full plasmid lists for the shortened cassettes shown in this graphic are presented in Section 5.21.

(irregular growth on the outside of the colony). It is therefore highly desirable that colonies display healthy growth when screening for reaction diffusion patterns, to not confuse these gene expression holes with a reaction diffusion pattern. Apart from general colony health, no trend was identified to produce a certain morphology. Wedges or the holes appearing in the red channel seem to appear randomly, and may be due to stochastic effects and biological variation from colony to colony.

2.4.3 Control 2: Node A Deletion

In this control, node A was removed from the full circuit, resulting in the circuit shown in Figure 25. This also removed *rpaI* from the circuit, which produces pC-HSL; thus only a single diffusor was present in the circuit (Du et al., 2020). Under classical Turing theory, this circuit should not produce any Turing patterns, even with a diffusor present.

This circuit was screened under a variety of media and agar thickness conditions. Since *tetR* was not present in the circuit, ATC should have no effect. Extensive screens showed that the circuit produced concentric rings in several different conditions. Figure 26 shows several example colonies of the typical observed morphology that produced thin green stripes on the outside of a predominantly red colony. Interestingly, this morphology resembles earlier timepoints of the irreproducible results from initial screens of the full circuit, as shown in Figure 20. Indeed, a side by side comparison between two of these colonies after roughly four

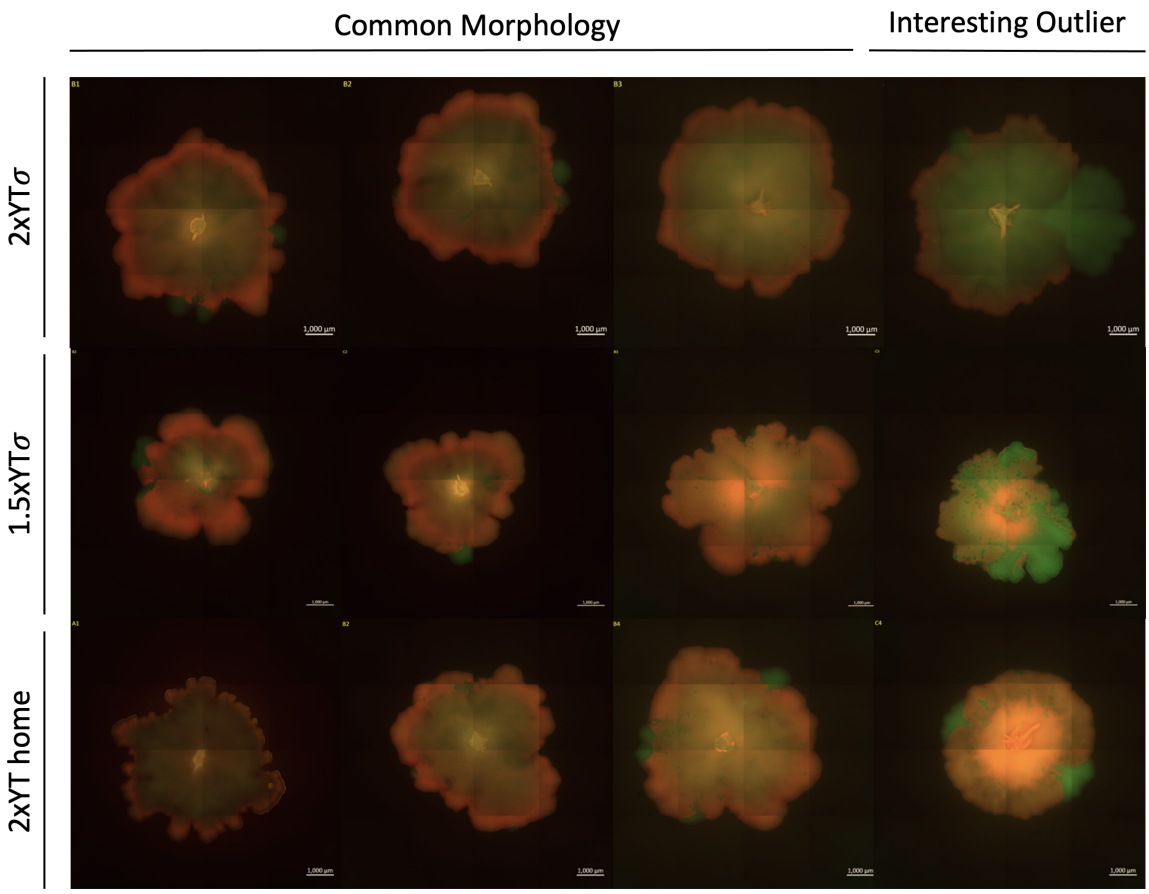


Figure 24: Compilation of different conditions for circuit variation 1, shown in Figure 23. Images were taken approximately 4 days after seeding on a wide field microscope; setup details are shown in Section 5.15. Agar thickness and media type do not appear to have an effect on morphology. Scale bar: 1 mm.

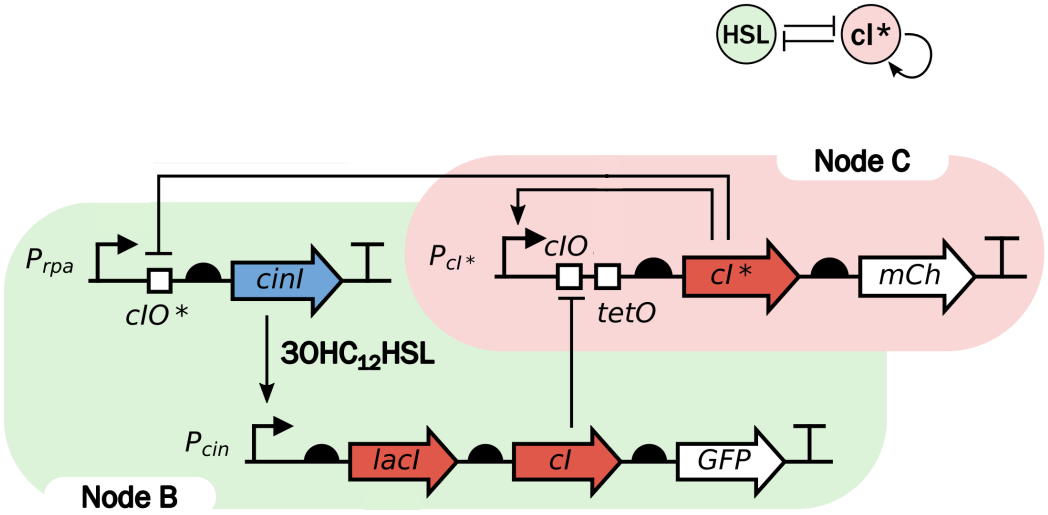


Figure 25: Circuit variation that does not contain node A. Deleting node A also removes the pC-HSL production enzyme *rpaI*. A schematic representation of the circuit is shown in the top right. Full plasmid lists for the shortened cassettes shown in this graphic are presented in Section 5.21.

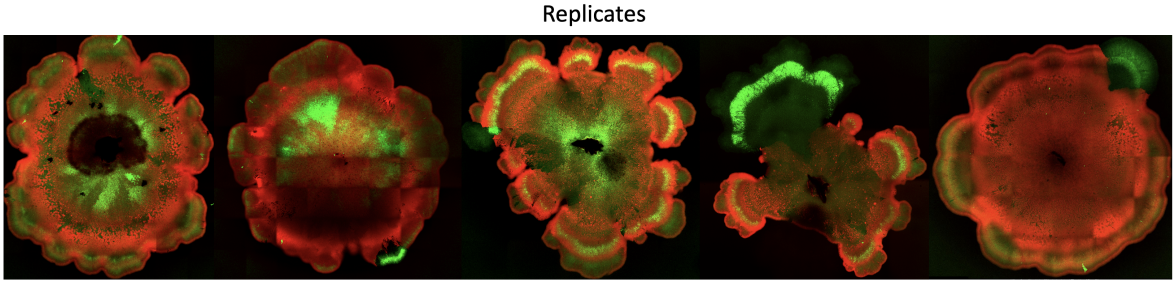
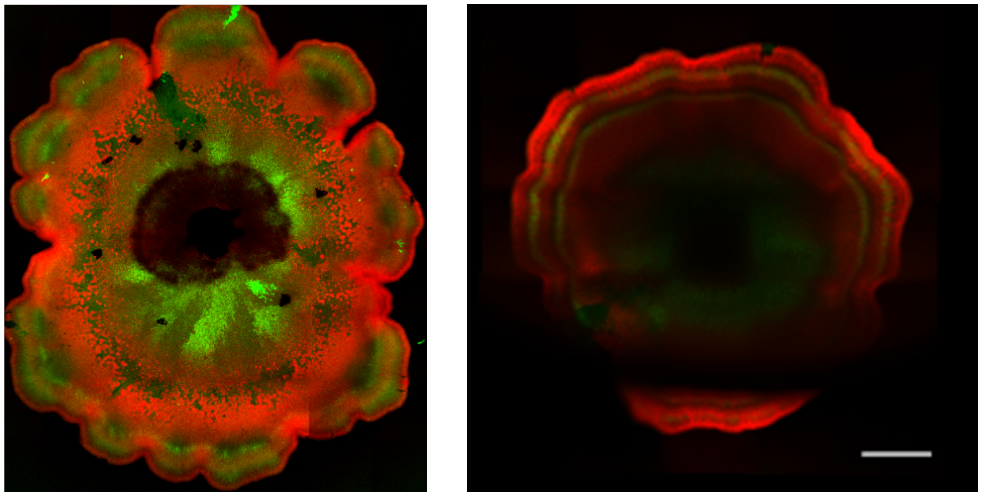


Figure 26: Compilation of patterning colonies observed containing circuit variation 2, a circuit schematic of which is shown in Figure 25. Images were taken approximately 4 days after seeding on a confocal microscope. Setup details are shown in Section 5.15



(a) Circuit Variation 2 (b) Full Circuit

Figure 27: Side by side comparison of a colony of circuit variation 2 (Figure 25) and the full circuit (Figure 12) imaged by Zhu (2020, unpublished results), which was not reproducible in the same conditions. Both images were taken approximately 4 days after seeding the colony; setup details are shown in Section 5.15. Scale bar: 1 mm.

days of growth highlights several common characteristics. Most notably, both colonies had two concentric green stripes, which were antiphase in an otherwise predominantly red region. Furthermore, both colonies were green in the centre, had a leading red edge, and had a hole of no gene expression in the middle.

2.5 Considering Turing Instability and Growth

As described in Section 1.2.3, growth can have an impact on both the shape and robustness of Turing patterns. Microscopy of the full circuit shown in Figure 20 indicates that growth may also have an impact on pattern formation. This is most apparent in Figure 20b, where the colony seems to have only grown in one direction from where it was seeded, and only the side which had faster growth was able to form patterns.

Since classical dispersion relation analysis cannot account for growth, a specialised dispersion analysis has to be implemented that can take growth into account. For this purpose, I adapted the dispersion analysis proposed by Madzvamuse et al. (2010). As discussed in Section 1.2.3, this analysis comes with several drawbacks. Firstly, it assumes isotropic[1], exponential[2] and dilational[3] growth, which may not faithfully capture all biological phenomena. Even with these assumptions, this analysis can still break down in some cases (Klika and Gaffney, 2017). However, unlike other methods, this type of analysis has the advantage that it does not require any extensive numerical analysis, and is computationally cheap, making it a good candidate for extensive parameter searches. In order to analyse the model shown in Equation 10, I extended the analysis from Madzvamuse et al. (2010) to account for 3 equation systems. Figure 28 shows a comparison between the expanded theory versus the original reported results when analysing the parameterspace of a Schnakenberg model, shown in Equation 16 (Schnakenberg, 1979; Madzvamuse et al., 2010). These results already show how the parameterspace can change (and increase) with increasing growth rate. Notably, some parameters that yielded Turing instabilities in dispersion relation analysis with higher growth rates did not produce patterns at lower growth rates and vice versa. This is already reminiscent of the results shown in Figure 20b, where slower growth rates did not yield a pattern.

$$\underbrace{\frac{\partial u}{\partial t}}_{\text{Rate of Change}} = \underbrace{D_u \nabla^2 u}_{\text{Diffusion}} + \underbrace{\gamma(a - u + u^2 v)}_{\text{Activator reaction}} \tag{16a}$$

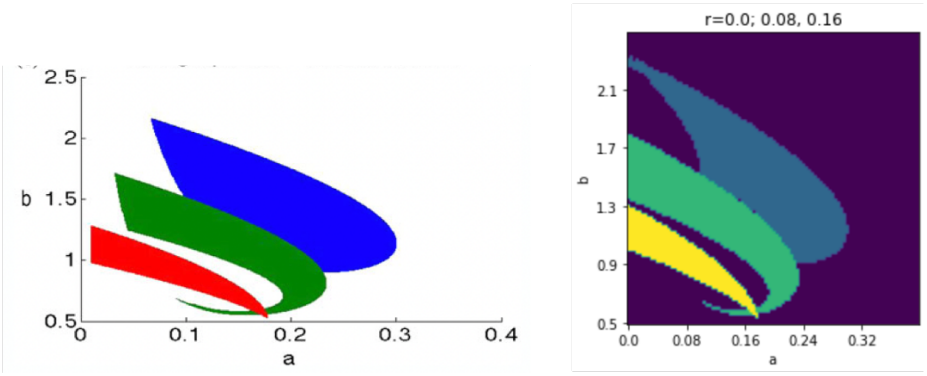
$$\underbrace{\frac{\partial v}{\partial t}}_{\text{Rate of Change}} = \underbrace{D_v \nabla^2 v}_{\text{Diffusion}} + \underbrace{\gamma(b - u^2 v)}_{\text{Inhibitor reaction}} \tag{16b}$$

After expanding the theory proposed by Madzvamuse et al. (2010) to three equations, the impact of growth on Turing pattern formation for the proposed circuit (Equation 10) was assessed. The adapted dispersion relation method is outlined in Section 5.2. During a parameter scan, 200,000 parameters were tested for different growth rates, and two useful

[1]Isotropic growth assumes two things: firstly, the area grows in each direction with equal speed; secondly, any two points placed on the growing surface will move apart with a speed based on their distance. Mathematically, this type of growth can be defined as equally increasing the Δx between each point on a discretised lattice during PDE solving. For more information see Section 5.3.

[2]Exponential growth assumes that the reaction space will grow indefinitely. This is not true for the modelled biological system (a colony), since the growth curve for bacterial growth is typically sigmoidal (if no bacteria are removed and the nutrient source is depleted over time).

[3]Dilational growth assumes that the number of molecules stays the same as the region expands. For example, assuming no reaction at all, an initial concentration of 1 molecule per cm^2 will dilute to 0.5 molecules per cm^2 as the area doubles in size.



(a) Original results (b) Generalised adaptation

Figure 28: Modified stability analysis that includes growth, using the Schnakenberg model (Equation 16) as an example (Schnakenberg, 1979). Figure 28a shows the original results obtained by Madzvamuse et al. (2010) when developing the method. Figure 28b shows a generalised adaptation of Madzvamuse et al. (2010). Both models assume two dimensional, exponential growth with growth rates of 0, 0.08 and 0.16. The parameter space of a and b that is predicted to form Turing patterns is increasing with increasing growth rates.

categories are highlighted: Firstly, Turing parameters, which are stable without diffusion ($k = 0$), have an eigenvalue $\lambda > 0$ for $k > 0$, and are stable ($\lambda < 0$) as $k \to \infty$. Secondly, "Peaks", which are stable throughout, but have a dispersion peak below 0 for $k > 0$, which returns back below the maximum peak value as $k \to \infty$. These are included, since with stochastic Turing theory, these peaks may be amplified into a stable pattern (Butler and Goldenfeld, 2009; Biancalani et al., 2017; Karig et al., 2018). Since the focus lies on finding a stationary pattern, both peaks (above and below zero) only get counted in the statistics of Figure 29 (and subsequent Figures in this chapter) if the imaginary part of the eigenvalue at the dominant wavelength is zero. As seen in Figure 29, no clear trend emerged for parametric robustness when inducing Turing patterns at varying growth rates for Equations 10. This was not the case for peaks below zero, which may lead to stochastic Turing pattern formation, where parametric robustness improved steadily with increasing growth rates.

While including growth for the full circuit (Equation 10) did not increase parameter space, as it did for the Schnakenberg model, certain parameters are only able to produce Turing patterns with the inclusion of growth (Schnakenberg, 1979; Madzvamuse et al., 2010). To illustrate this, five different growth rates r were used as categories for a Venn diagram (Figure 30). Two diagrams were generated; one for Turing instabilities, where the dispersion peak was above zero, and one for dispersion peaks below zero. The numbers in the Venn diagram show the percentage of the total number of Turing instabilities (Figure 30a) or a percentage of the total number of dispersion peaks below zero (Figure 30b). Interestingly, although growth did not have an impact on parametric robustness when it came to Turing instabilities, less

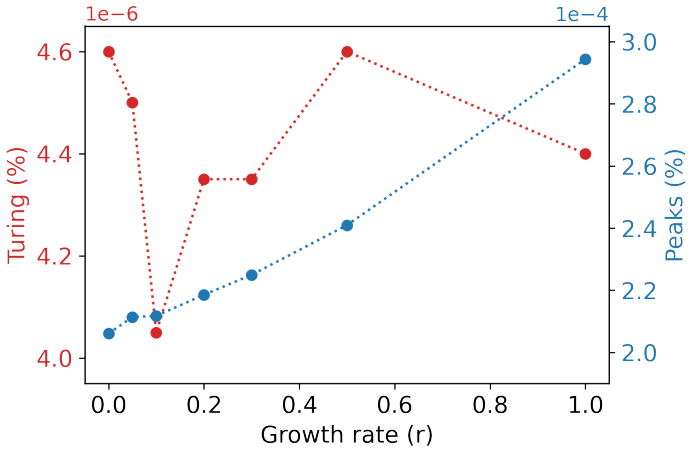
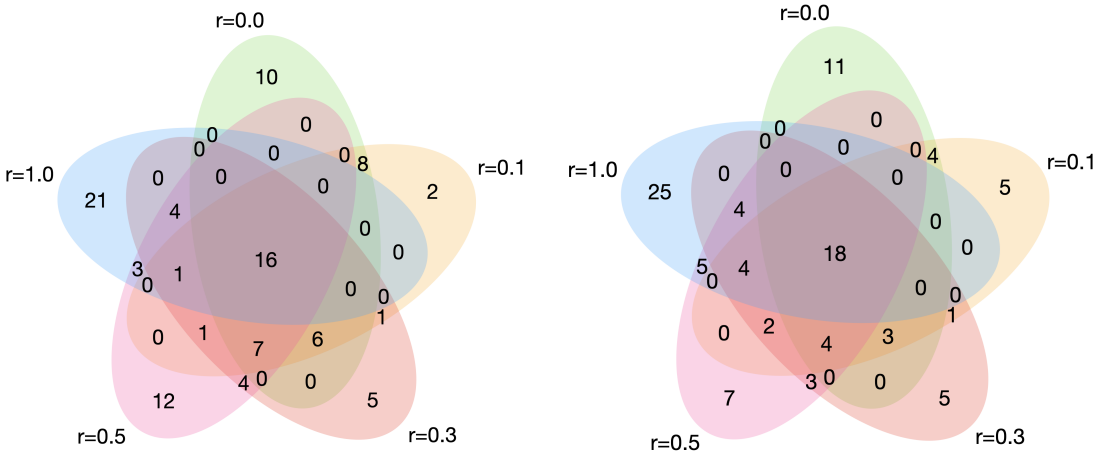


Figure 29: Percentages of found Turing instabilities (red dots) and dispersion peaks below 0 (blue dots) from a large parameter scan (200,000 sampled parameters) at different growth rates (r). No clear trend emerges for Turing instabilities, while parametric robustness increases for dispersion peaks below zero at increasing growth rates. The stability analysis used is adapted from Madzvamuse et al. (2010), and described in Section 5.2.

than 50% of the parameters sampled yielded Turing instabilities without the inclusion of growth. Moreover, ca. 40% of Turing parameters were solely present at a specific growth rate, and only ca. 16% yielded a Turing instability at any of the sampled growth rates. This is also reflected in several microscopy images. Firstly this may explain the observation of Figure 20b, where the pattern only formed on the faster growing side of the colony. Secondly, this shows how the colour shift control shown in Figure 21 was possible. Even thought the full circuit was imaged in conditions that usually yield at least some patterning, pattern formation was disrupted by supplying richer media, where colonies grew more quickly. By increasing growth rate, the colonies were no longer in the patterning regime, and the response of the whole circuit to ATC could be observed.

In addition to analytical analysis, numerical simulations were performed to check the impact of growth and curvature on the full circuit when in a Turing parameter set. Figure 31 shows different behaviours of a simulated growing colony at three different growth rates. At slow growth, rings were added on the inside of the colony, and the stripes moved as the colony expanded. At intermediate growth, a labyrinthine pattern formed. At fast growth rates, rings were added on the outside of the growing colony. This matches experimental data in a chemical system. In the chemical setup, the outside of an area of interest was inhibited. The reaction area was then expanded at different rates, and the behaviour was categorised (Konow et al., 2019).

Since the inclusion of growth in dispersion relation analysis has been shown to enable pat-



(a) Turing Instabilities (b) Dispersion Peaks

Figure 30: Venn diagrams of parameter searches for Turing instabilities at different growth rates. The figures show a percentage of how the total number of found Turing instabilities (Figure 30a) or dispersion peaks below zero (Figure 30b) are distributed. Percentages are rounded to the closest whole number. The stability analysis used is adapted from Madzvamuse et al. (2010), and described in Section 5.2.

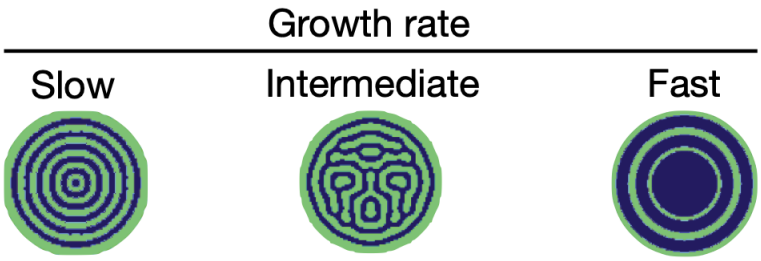


Figure 31: The same Turing parameter set is simulated in a growing colony. With slow growth, new rings are added in the centre, and a leading edge remains green. With intermediate growth, a labyrinthine pattern forms. With fast growth, rings are added on the outside. This matches experimental data published in a chemical system (Konow et al., 2019). Modelling based on Equation 10.

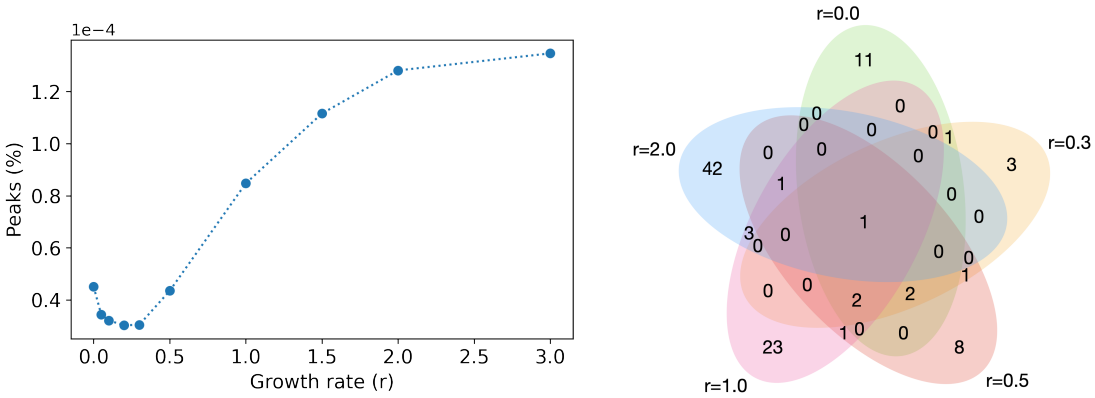
tern formation in topologies that are not able to form Turing patterns in classical dispersion relation analysis, two biologically observed phenomena were analysed with the consideration of growth. I first analysed circuit variation 2, which, as shown in Figures 26 and 27, formed green stripes on the outer edge of a predominantly red colony. This circuit variation was modelled by Equation 17. Despite searching ca. 2,000,000 parameter combinations, of which 1,800,000 include a growth rate $r > 0$, no parameter set was found that showed a dispersion peak above or below 0.

$$\frac{\partial[HSL]}{\partial t} = D_{HSL}\nabla^2[HSL] + b_{HSL} + V_{HSL}\frac{1}{1+(\frac{[cI*]}{k_{A_{cI_1^*}}})^{n_{cI_1^*}}} - \mu_{HSL}[HSL] \tag{17a}$$

$$\frac{\partial[cI]}{\partial t} = b_{cI} + V_{cI}\frac{(\frac{[HSL]}{k_{A_{HSL}}})^{n_{HSL}}}{1+(\frac{[HSL]}{k_{A_{HSL}}})^{n_{HSL}}} - \mu_{cI}[cI] \tag{17b}$$

$$\frac{\partial[cI*]}{\partial t} = V_{cI*}\frac{1}{1+(\frac{[cI]}{k_{A_{cI}}})^{n_{cI}}}\frac{(\frac{[cI*]}{k_{A_{cI_2^*}}})^{n_{cI_2^*}}}{1+(\frac{[cI*]}{k_{A_{cI_2^*}}})^{n_{cI_2^*}}} - \mu_{cI*}[cI*] \tag{17c}$$

The second analysed subcircuit is an apparent loss of node C from the full circuit. A circuit schematic for this subcircuit is shown in Figure 34a. This subcircuit was analysed because of the green wedges that have lost all expression potential for *mCherry*, and still showed repeated patterns in the green channel. This is particularly apparent in the top of Figure 20a, where two of such wedges appeared. In the repeat of said results (Figure 20b), another such wedge began to form in the bottom right of the colony. One explanation for this wedge formation is that the plasmid carrying node C was lost as the colony grew. The plasmid is maintained via Ampicillin resistance. Since Ampicillin is degraded extracellularly, it is possible that the antibiotic is inactivated at later time points (Green, 2012). To model this subcircuit resulting from plasmid loss, Equation 18 was used, and analysed with the modified stability analysis to take growth into account (Section 5.2) (Madzvamuse et al., 2010). Despite analysing ca. 2,000,000 parameter combinations, 1,800,000 of which included growth, no parameter set was found that showed signs of classical or growth induced Turing instability. However, several parameter sets were found with a dispersion peak below zero, which can lead to patterning according to stochastic Turing theory (Butler and Goldenfeld, 2009; Biancalani et al., 2017; Karig et al., 2018). Additionally, growth again had a positive impact on inducing dispersion peaks below zero (Figure 32a). As can be seen in Figure 32b, very few parameters showed a dispersion peak below zero for multiple growth rates.



(a) Trendline (b) Venn Diagram

Figure 32: Impact of growth on the abundance of dispersion peaks found below zero for the full circuit without node C. 200,000 parameters were sampled at different growth rates (r). The stability analysis used is adapted from Madzvamuse et al. (2010) and is described in Section 5.2. Figure 32a shows that parametric robustness increases for dispersion peaks below zero at increasing growth rates. Figure 32b shows the percentage of the total number of found dispersion peaks below zero.

An example of a dispersion peak below zero that was induced via growth is shown in Figure 33. To further determine whether the stripe formation is due to complete plasmid loss or due to some other unknown phenomenon, the full circuit without node C (Equation 18) was transformed in *E. coli* and imaged with confocal microscopy. As seen in the results in Figure 34b, the subcircuit indeed formed repeated green stripes. Because node C is usually the carrier of *mCherry*, no red fluorescence is present in the image.

$$\frac{\partial[pC]}{\partial t} = b_{pC} + V_{pC}\frac{1}{1+(\frac{[lacI]}{k_{A_{pC}}})^{n_{pc}}} - \mu_{pC}[pC] + D_{pC}\nabla^2[pC] \tag{18a}$$

$$\frac{\partial[HSL]}{\partial t} = b_{HSL} + V_{HSL}\frac{(\frac{[pC]}{k_{A_{HSL}}})^{n_{HSL}}}{1+(\frac{[pC]}{k_{A_{HSL}}})^{n_{HSL}}} - \mu_{HSL}[HSL] + D_{HSL}\nabla^2[HSL] \tag{18b}$$

$$\frac{\partial[lacI]}{\partial t} = b_{lacI} + V_{lacI}\frac{(\frac{[HSL]}{k_{A_{lacI}}})^{n_{lacI}}}{1+(\frac{[HSL]}{k_{A_{lacI}}})^{n_{lacI}}} - \mu_{lacI}[lacI] \tag{18c}$$

Circuit variation 1 (Section 2.4.2) was not analysed via dispersion relation analysis. Since no diffusors act on the circuit, the characteristic equation does not depend on a wavenumber k, and getting a dispersion peak becomes impossible.

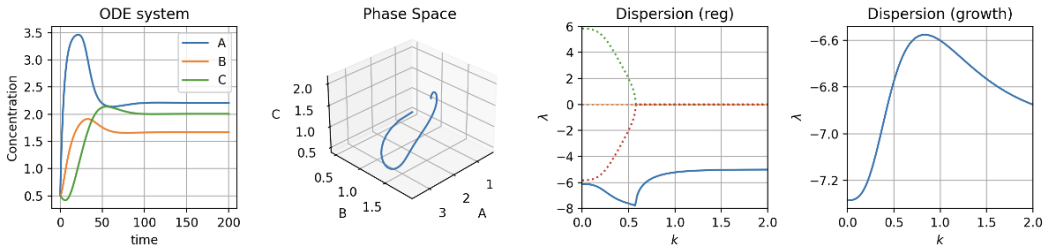
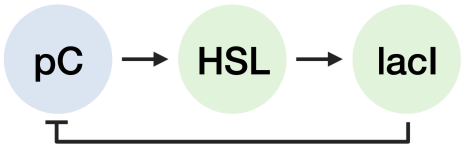
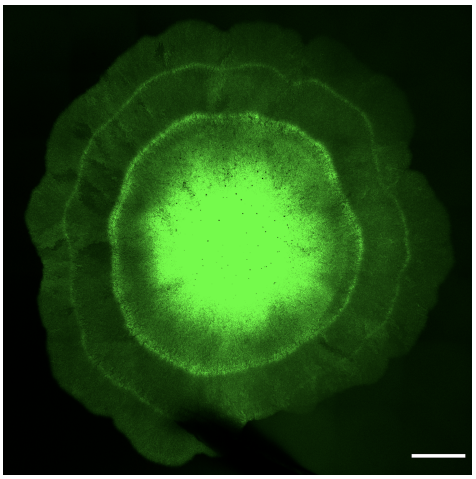


Figure 33: Linear stability analysis of Equation 18 when growth is considered. Figures from left to right show how the well mixed system evolves to steady state, the phase space of how the system reaches steady state, the dispersion plot without growth, and the dispersion relation when including growth. The solid blue lines in the dispersion plots are the maximum of the real parts of the eigenvalues. Dashed lines show the imaginary parts.



(a) Circuit schematic



(b) Confocal microscopy

Figure 34: Figure 34a shows the circuit schematic of the participating genes in the full circuit without node C. 3OHC$_{14}$HSL is abbreviated as HSL, and pC-HSL as pC. The subcircuit can be described by Equation 18. Figure 34b shows confocal microscopy of the full circuit without node C, which normally carries *mCherry*. The image shows two repeated stripes in the green channel. Microscopy image obtained by Jure Tica (2023, unpublished results). Scale bar: 1 mm.

2.6 Discussion & Conclusion: Engineering Turing Patterns

In this chapter, *E. coli* were used to study synthetic pattern generation with a reaction-diffusion mechanism. For this purpose, a robust Turing network based on circuit 3954 from Scholes et al. (2019) was implemented and tested. Initial screens in liquid culture showed that the circuit connections appeared to be intact, and were functioning as predicted (Section 2.2). Testing the effect of the diffusive molecules showed that diffusion was slow enough that a diffusion profile could form, and that it could be measured via a fluorescent output. If diffusion were a lot faster than the kinetics of the circuit, the whole surface would have been induced in the experiments in Section 2.3, which would impede pattern formation.

Initial microscopy screens indeed resulted in a repeated stripe pattern in growing bacterial colonies. However, the pattern was not reproducible. In an attempt to debug the assay, the circuit was tested outside of the typical patterning regime, to see whether the circuit still responds to small molecules when imaged as a growing colony. This confirmed that the growing colonies are still tuneable via small molecules. A stringent parameter scan across multiple agar thicknesses, richnesses, and ATC concentrations revealed another patterning regime, where green spots appeared. Furthermore, the spots appeared to increase in size as the ATC concentration was increased.

To examine whether this pattern was due to a reaction diffusion mechanism or not, a control was extensively screened, where no quorum sensing molecule was linked to the circuit. Since no pattern was observed, the observed patterns were likely due to a reaction-diffusion mechanism. A second control was constructed that should not produce a classical Turing pattern, but includes a diffusing molecule linked to the circuit. This has the effect, that cells that become stuck in a state that produces the small molecule can act as inducers for neighbouring cells, potentially leading to close proximity circuit state syncing. While analytical analysis revealed that this circuit should not produce classical Turing patterns, microscopy still showed concentric green rings forming on the outside of a growing colony.

Since microscopy revealed that growth has a large impact on pattern formation in this system, extensive modelling was employed to investigate the effect of growth on pattern formation and parametric robustness. Furthermore, growth can induce pattern formation in Turing patterns (both by increasing parameter space, as well as by enabling novel topologies, including only using a single diffusor) (Madzvamuse et al., 2010; Wang et al., 2022). Including growth during modelling could not explain how this pattern could form according to conventional Turing theory. However, extensive modelling of both the full circuit, as well as other subcircuits, revealed that growth increases the abundance of dispersion peaks below zero -

which have been shown to amplify into a stochastic Turing pattern in some cases (Butler and Goldenfeld, 2009; Biancalani et al., 2017; Karig et al., 2018). To test this finding, one subcircuit (shown in Figure 34a) was imaged, which should not be able to produce classical Turing patterns. However, the subcircuit had an increasing likelihood of producing stochastic Turing patterns at increased growth rates. As shown in Figure 34b, this subcircuit was indeed able to produce repeated stripe patterns.

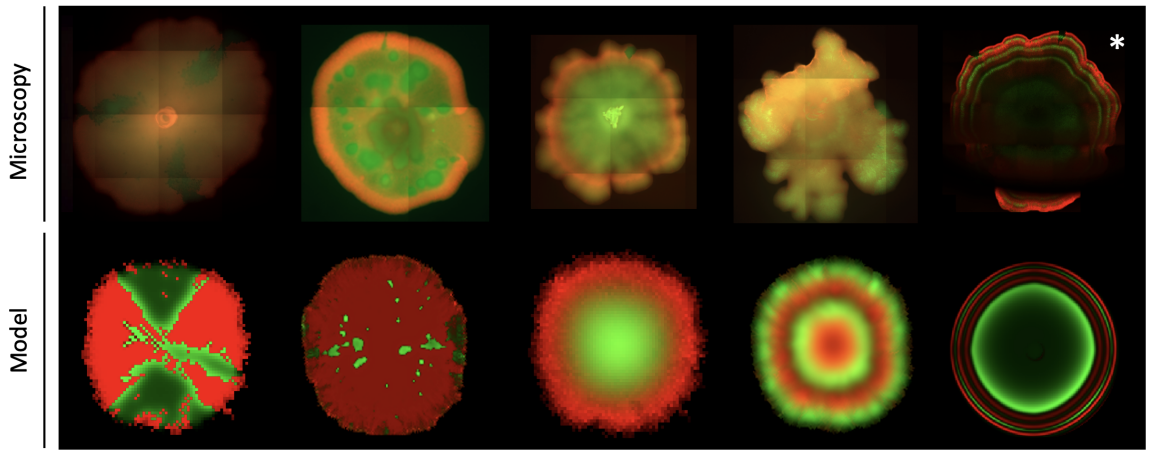


Figure 35: Compilation of different observed gene expression patterns compared to a cellular automata model. All parameter sets apart from the first image display Turing instability measured by dispersion relation analysis. The conditions for the microscopy figures were as follows from left to right: 0.75 ml 2xYT medium, 1.4% agar, 15 µM ATC. 0.75 ml 2xYT medium, 1.4% agar, 30 µM ATC. 0.75 ml 2xYT medium, 1.4% agar, 5 µM ATC. 0.75 ml 2xYT medium, 1.4% agar, 15 µM ATC. 1 ml 2xYT medium, 1.4% agar, 10 µM ATC. The final microscopy image (marked by an asterisk) was imaged via confocal microscopy by Tong Zhu in a 6 well tissue culture plate (2020, unpublished results). All other images were obtained via wide field microscopy as described in Section 5.15. All modelling in this figure was performed by Martina Oliver-Huidobro (2023, unpublished results).

While the definitive cause of patterning is still difficult to determine, the engineered circuit is undoubtedly a diverse pattern generator with some tuneability. A compilation over multiple patterning regimes is shown in Figure 35. Notably, the modelled patterns are due to an underlying Turing mechanism. This suggests that the engineered Turing mechanism may be the cause for symmetry breaking and pattern formation in at least some of the observed colonies. Because the Turing mechanism alone is not enough to explain all the patterning presented in this chapter, it is also apparent that we still have a lot to learn about patterning mechanisms. While novel analytical methods are able to explain and predict some patterning, patterns still appear more frequently in biology than expected - even in a system that is engineered from the ground up using well understood biological parts. This inability to reconcile some of the observed patterns may be due to two factors: Firstly, because of inefficient theo-

retical understanding of patterning mechanisms, and an oversimplification of the circuit. For instance, all modelling in this thesis relies on PDEs, which assume an uninterrupted reaction-diffusion space. In biology, cells are separate entities. Some modern modelling frameworks take this into account, and model similar biological systems with *cellular automata*, which are spatially separated (Tordoff et al., 2021; Mordvintsev et al., 2020). The second reason is that parts used in synthetic biology may sometimes fail in unforeseen ways, even when they are well characterised. An example of this is the spontaneous failure of parts using the Cello framework, where all parts and gates work well independently, but then single parts, as well as whole gates fail when they are put together (Espah Borujeni et al., 2020). Better theoretical understanding of which processes are indispensable, as well as which processes can be simplified, alongside more robust biological parts could help to close the gap between the frequency of observed and expected patterns in engineered biological systems.

3

Pattern Recognition: Neuromorphic Computing

Apart from pattern generation, gene expression pattern recognition is needed to enable macro level design of biology. In this Chapter, I use neuromorphic computing devices to mimic the nodes of an artificial neural network via sequestration reactions. I first confirm that endoribonucleases (ERNs) of the PERSIST platform can be used to build individual molecular neurons. I then enable the tuneability of these nodes via molecular weights and biases; the prerequisites to build a general purpose neural network. To confirm these individual nodes can be tuned and used with one another, I build two molecular neural networks - one forming a two input dual region, and the other a two input bandpass filter. Finally, I introduce a weight tuning strategy where molecular weights can be permanently adjusted in an inducible manner.

3.1 Building Molecular Perceptrons

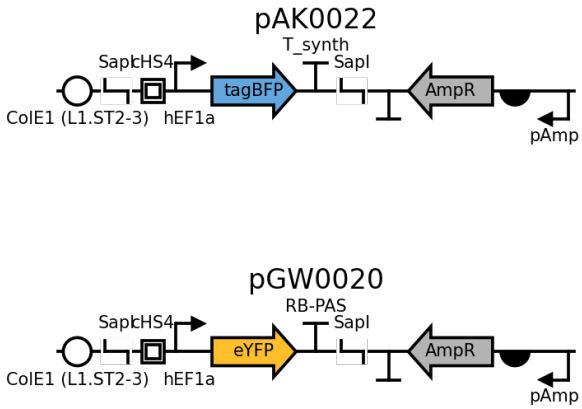
As introduced in Section 1.3.3, the most straight forward approach to build a molecular neural network is via a sequestration mechanism. One such mechanism is the PERSIST mechanism (shown in Figure 11), where one ERN-hairpin pair can function as a two-input perceptron. In order to confirm that this mechanism indeed behaves like a perceptron, its dose response has to be tested. Furthermore, a method needs to be established, that can robustly yield 4 dimensional dose responses. (For a full perceptron, two fluorescent markers are used up by the inputs, one by the bias, and one by the output.) I explored poly-transfection for this purpose, which allows for multidimensional dose response testing in a single well (Gam et al., 2019).

In poly-transfection, the stoichiometry of standard mammalian transfection is exploited, to give high resolution dose responses in multiple dimensions. This high dimensionality is achieved via controlled correlation and decorrelation of individual plasmids. For instance, imagine plasmid A expresses an ERN and serves as a negative input to a perceptron, and

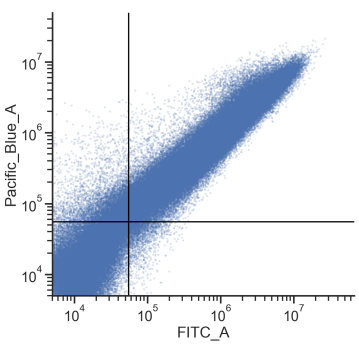
plasmid B serves as a positive input to a sequestron (e.g. plasmid B expresses fluorescent protein O, whose expression is repressed by A). If A and B are both correlated to a specific colour, but are purposefully uncorrelated from one another, a wide expression spectrum of A vs B can be achieved. This effect of controlled correlation and decorrelation is showcased in Figure 36b, where Figure 36b shows purposeful correlation, and Figure 36c shows purposeful decorrelation. By measuring the output O, and knowing the amount of A and B, the effect of A on B can be quantified. Figure 36b shows how two plasmids can be linearly correlated (in this case in a 1:1 co-transfection ratio), and Figure 36c shows how a wide spectrum of expression ratios can be achieved via poly-transfection (for method details, see Section 5.4. Since each single cell can now contain a random ratio of each input, and each input is linearly correlated with a measurable colour, flow cytometry can be used to determine the input and output levels for each cell separately (Gam et al., 2019). For method details see Sections 5.5 and 5.6.

I screened three orthogonal ERNs for their ability to act as molecular perceptrons, as described in Section 1.3.3: *CasE*, *Csy4* and *pguCas13b*. Each ERN was co-tranfected, and thus linearly correlated with *mKate2*. Each ERN sequesters a target mRNA molecule expressing *eYFPG5A*. *eYFPG5A* was co-transfected with *tagBFP*. Using poly-transfection, the two inputs (x_1 being the *tagBFP* and *eYFPG5A* co-transfection, and x_2 being the *mKate2* and ERN co-transfection) can be decorrelated to produce a detailed dose response in the *eYFPG5A* channel. The layout of these three molecular perceptrons can be found in Figure 37. A detailed plasmid map for all 8 plasmids used in this experiment can be found in Section 5.21.

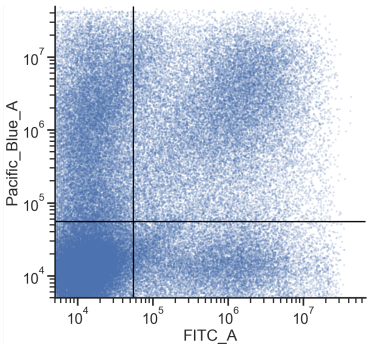
By definition, input x_1 is positively weighted, and input x_2 is negatively weighted. The exact weight depends on several biological factors, such as transcription and translation efficiency, and the efficiency of the sequestration reaction. In theory, the weight can be easily tuned by changing the ratio between the transfection marker and the active component (in this case *tagBFP* for *eYFPG5A* with the ERN recognition site, and *mKate2* for the respective ERNs). Characterising each perceptron implementation via poly-transfection results in Figure 38. As can be seen, while the different ERNs have different innate strengths, the resulting heatmaps display the desired perceptron behaviour that was introduced in Section 1.3.2. This mechanism could therefore serve as a starting point to implement molecular neural networks in mammalian cells.



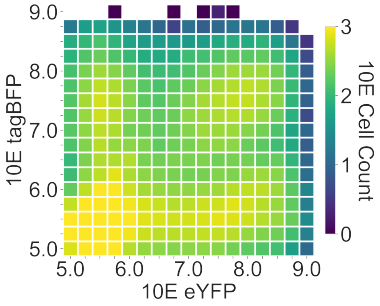
(a) Plasmids used



(b) Co-transfection



(c) Poly-transfection



(d) Cell count

Figure 36: Co-transfection and poly-transfection examples. Figure 36a shows the constitutive colour expression plasmids used. pAK0022 expresses *tagBFP* and pGW0020 expresses *eYFP*. The FACS channel Pacific_Blue_A measures the response of a constitutively expressed *tagBFP* protein, and the FITC_A channel measures the expression of *eYFP*. Flow cytometry data in Figures 36b, 36c and 36d are obtained using the method described in Section 5.5, and analysed using the pipeline described in Section 5.6. Figure 36b shows how two colours are linearly correlated when packaged in a co-transfection in a 1:1 ratio. Figure 36c shows the wide range of expression ratios that are achieved via poly-transfection. Figure 36d shows a processed version of Figure 36c, where each pixel is coloured based on how many cells are in each pixel. Distributions are similar for all transfections shown in Section 3. For transfection, flow cytometry, and analysis details see Sections 5.4, 5.5, and 5.6 respectively.

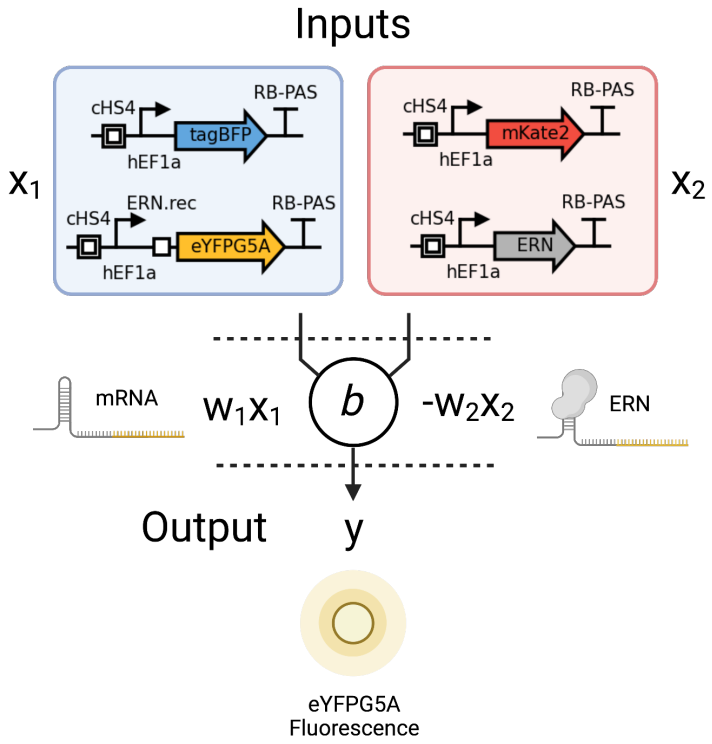
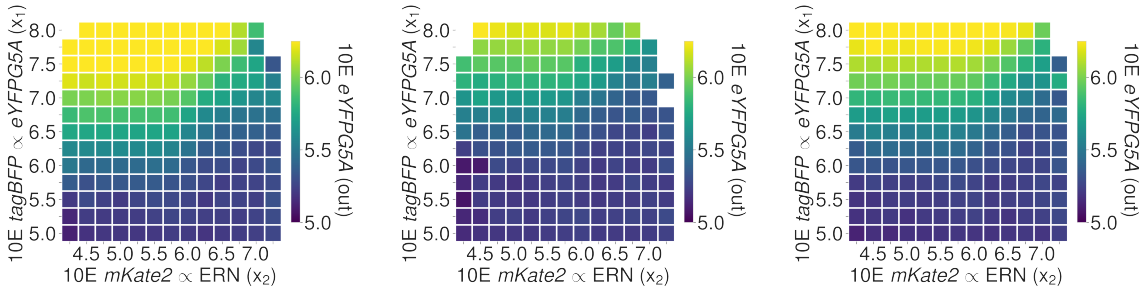
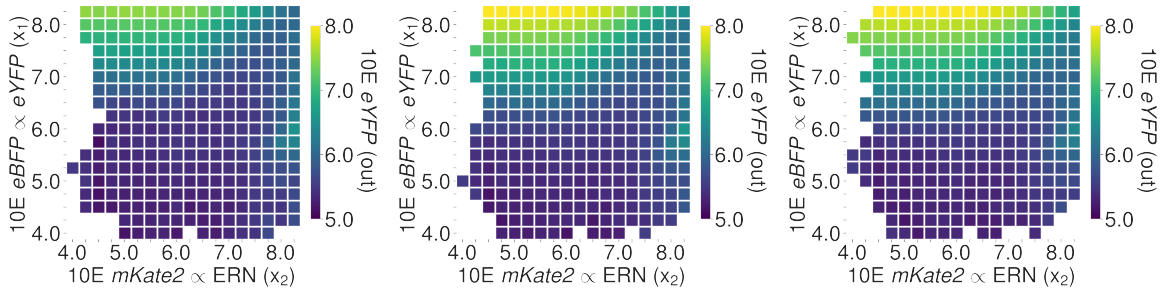


Figure 37: Setup to screen for perceptron behaviour. *tagBFP* and and *eYFPG5A* are co-transfected, and are thus correlated linearly (as seen in Figure 36b). *eYFPG5A* contains an ERN recognition site, and is sequestered at the RNA level by an ERN. The ERN is co-transfected (and linearly correlated) with *mKate2*. Three ERNs are tested in this way, together with their recognition site: *CasE* (ERN plasmid pGW0100, ERN rocognition site plasmid pGW0073), *Csy4* (ERN plasmid pGW0101, ERN rocognition site plasmid pGW0103), and *pguCas13b* (ERN plasmid pGW0102, ERN rocognition site plasmid pGW0104). Detailed plasmid maps can be found in Section 5.21.



(a) *CasE* Perceptron (b) *Csy4* Perceptron (c) *pguCas13b* Perceptron

Figure 38: Flow cytometry results of ERN based molecular perceptrons. The positively weighted input (x_1) is increasing logarithmically along the y-axis, while the negatively weighted input (x_2) is increasing logarithmically along the x-axis. The colourbar indicates the measured response (expressed *eYFPG5A*). Each pixel is coloured based on the mean of all cells that are encountered in that particular bin of *mKate2* and *tagBFP*. If a pixel contains less than 12 cells, the pixel is coloured white. For analysis details see Section 5.6.



(a) $w_1 = 0.5$, $w_2 = -1$ (b) $w_1 = 1$, $w_2 = -1$ (c) $w_1 = 1$, $w_2 = -0.5$

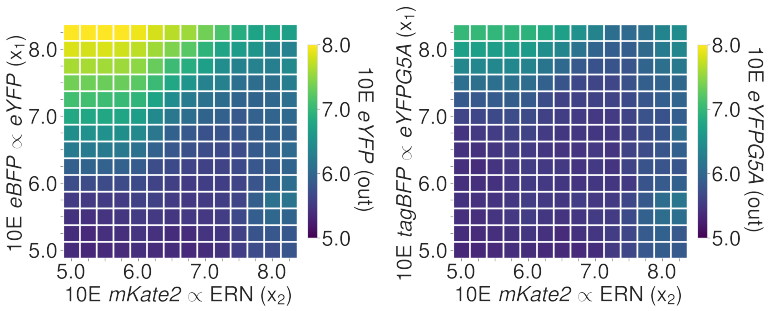
Figure 39: Different co-transfection ratios are used to tune a CasE based molecular perceptron. The perceptron layout used is based on Figure 37, where w_1 is associated with the output potential (*eYFPG5A*), and w_2 is associated with the amount of ERN (in this case *CasE*) in the system. Based on the weights, the overall signal should increase from left to right. Data representation is consistent with Figure 38. The plasmids used in this experiment are pNW114.001 (for *eYFP*), pAK0007 (for *eBFP*), pSEP1.08 (for *CasE*), and pAK0006 (for *mKate2*), a detailed map of which can be found in Section 5.21. For analysis details see Section 5.6.

3.2 Designing a Library of Biological Weights

As shown in Section 3.1, the PERSIST platform can be used to build a molecular perceptron with both positive and negative inputs. However, artificial neural networks also require tuneable weights between the inputs, the different nodes, and the output(s). Without tuneable weights, artificial neural networks would not be able to produce different classification tasks while maintaining the same network architecture, fine-tuning would not be possible, and the network would not be able to hold memory (Abiodun et al., 2019; Emmert-Streib et al., 2020; Schuman et al., 2022; Racovita et al., 2022).

For the purpose of this thesis, I separated biological weight tuning strategies into two general categories: Permanent and transient. Transient weight tuning strategies allow for weight changing, but the weight is not remembered by the circuit. The quickest way to transiently lay out a molecular neural network with different weights is by changing the co-transfection ratios between components. For instance, a 1:1 ratio between the transfection marker for the ERN produces a set amount of protein, and produces a certain output, while a 1:2 marker:ERN ratio produces more ERN (x_2) has a higher negative weight); therefore the final output is lower. An example of how transient transfection affects a single molecular perceptron that uses *CasE* is shown in Figure 39.

While transient weight tuning is useful for prototyping, long term applications (such as cell therapies) need a permanent solution. Permanent weight tuning strategies need to not only store weights, but also remember and maintain that weight over multiple cell generations.



(a) Without uORF (b) With uORF

Figure 40: Dose response of a *CasE* based molecular perceptron. Figure 40a shows the response without a uORF. Figure 40b shows the reduced expression potential of the output based on the upstream uORF. Plasmid pGW0010 abundance is shown on the x-axis for both subfigures. Plasmid pGW0042 and pGW0079 abundance is shown on the y-axis for Figure 40a and Figure 40b respectively. Full plasmid maps are shown in Section 5.21. For analysis details see Section 5.6.

For mammalian cells, this means that weights need to be stored in genetic components that are able to be integrated into the genome. One known strategy for this is by using upstream open reading frames (uORFs) upstream of the gene of interest in order to tune down its transcription efficiency (Ferreira et al., 2013). The uORFs I use in this thesis are 12 base pairs long. The first three nucleotides are the last three nucleotides of the Kozak sequence (`ACC`), in order to start transcription initiation. These are followed by a start codon (`ATG`), in order to start the reading frame, which is followed by a single triplet codon (`GGT` for glycine), and end in a `TGA` stop codon to end the reading frame (Alberts et al., 2008; Ferreira et al., 2013). Figure 40 shows the response of a *CasE* based molecular perceptron, where a uORF is integrated upstream of the output gene (*eYFPG5A*) for one of the plots. As seen, this reduces the expression of the output, changing the weight of x_1.

As an additional strategy, I attempted to engineer the binding affinity between the ERN and its recognition site. As described in Section 1.3.3, each ERN binds a specific RNA sequence, which is then cleaved. The incomplete RNA sequence is then degraded, and thereby protein production is inhibited. I attempted to alter the affinity between the ERN and its respective recognition site by modifying the targeted RNA sequence. To maintain simplicity, I only engineered *CasE* recognition sites.

Minimum free energy prediction revealed that the *CasE* recognition site is composed of one long stem loop, followed by a short turn (Lorenz et al., 2011). The predicted secondary structure for the wild type recognition site is shown in Figure 41. I used three different approaches to change the binding affinity between this RNA structure and *CasE*. An overview of all designs can be found in Table 3.

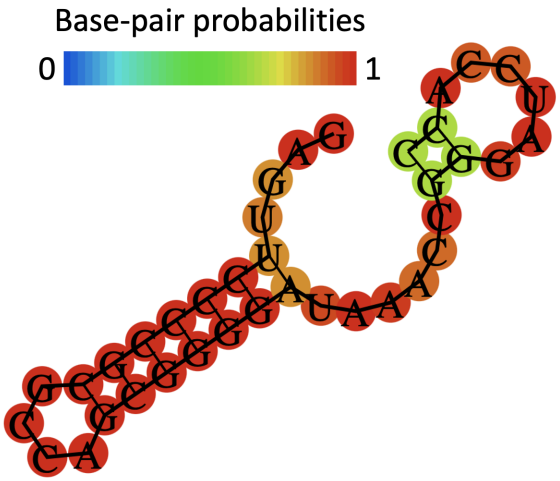


Figure 41: Secondary structure for the *CasE* recognition sequence, as modelled via free energy prediction by the Vienna RNA package (Lorenz et al., 2011). Base-pair probabilities change from blue to red.

CasE rec ID	**Design Strategy**	**ΔG (kcal/mol)**	**Stem Loop Frequency**	**Stem length (bp)**	**Sequence**
WT	0	-16.24	78.67%	7	gagttccccgcgccagcggggataaaccg ggatccacc
2	1	-12.94	73.52%	7	gagttcTTcgcgccagcgAAgataaaccg ggatccacc
3	1	-7.95	73.22%	7	gagttcTTTAcgccagTAAAgataaaccg ggatccacc
4	1	-3.64	45.62%	7	gagttTTTTATgccaATAAAAataaaccg ggatccacc
5	1	-9.54	65.01%	7	gagttccccgcgccagcgAggataaaccg ggatccacc
6	2	-5.49	42.61%	7	gagttccccgcgccagcgAAgataaaccg CCatccaGG
7	2	-7.71	29.83%	7	gagttccAcgcgccagcggTgataaaccg ggatccacc
8	2	-4.94	34.72%	7	gagttTcAcTcgccagcgTTgataaaccg ggatccacc
9	3	-12.94	73.68%	6	gagttcccgcgccagcgggataaaccg ggatccacc
10	3	-9.64	73.70%	5	gagttccgcgccagcggataaaccg ggatccacc

Table 3: Design strategy and specifications for *CasE* recognition site library. The long stem loop is shown in bold, which is followed by the short turn. The short turn is kept constant, unless not changing the turn sequence had a large impact on the predicted secondary structure. Capitalised letters denote a substitution from the wild type sequence. CasE.rec ID 8 serves as a negative control; the 34% predicted likelihood structure is a competing secondary structure that appears with higher frequency than the wild type structure.

In the first design strategy, I assumed that the secondary structure of the recognition site is absolutely essential for protein binding, but the sequence that produces this structure is less critical. In this case, I substituted strong `G-C` base pairs with weaker `A-T` base pairs, while I attempted to avoid altering the predicted frequency of the dominant (wild type) secondary structure. I assumed that the binding affinity would decrease with lower stored thermodynamic energy (less negative ΔG). This approach is labelled as design strategy 1 in Table 3.

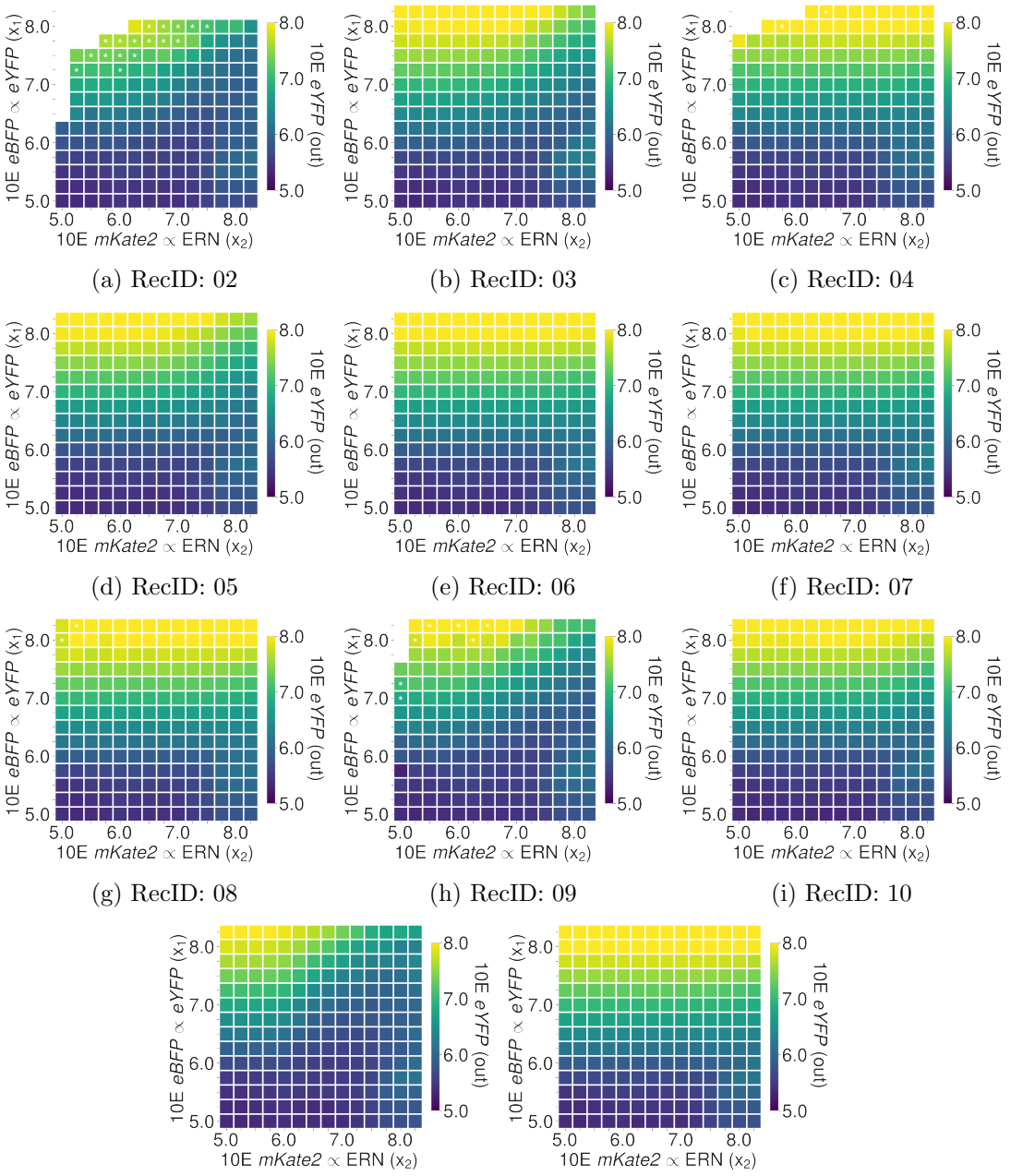
In the second design strategy, I assumed that small changes in secondary structure can be tolerated, but reduce the overall binding affinity of *CasE* to the recognition site. By allowing for slight changes in the overall secondary structure, I was able to introduce competing secondary structures. This changes the frequency at which the dominant secondary structure (that resembles the wild type structure) appears. I designed this strategy in case *CasE* binding is Boolean (either it binds and cuts the sequence, or it does not recognise it at all). My reasoning was that introducing a competing sequence would allow for a modified equilibrium between recognisable and unrecognisable RNA secondary structures. This would lead to an apparently decreasing binding affinity between different sequences. This strategy is labelled as design strategy 2 in Table 3.

For the final design strategy, I assumed that either introducing a competing secondary structure, or changing the ΔG per base in the stem loop prevents *CasE* from binding its recognition site. For this case, I decreased the stem loop length by up to two base pairs (compared to the 7 base pair long wild type), in case a shorter stem loop leads to decreased *CasE*-RNA binding affinity. This strategy is labelled as design strategy 3 in Table 3.

Figure 42 shows all screened RNA secondary structures in the context of a molecular perceptron. As seen in this Figure, five of the engineered *CasE* recognition sites (with the RecIDs 2, 3, 5, 9, and 10) show a reduced affinity for *CasE*, while still allowing for *CasE* binding.

3.3 Including a Biological Bias

Another important aspect of perceptrons is the inclusion of a bias. As mentioned in Section 1.3.2, the bias is a value that is added to the weighted sum of the inputs, before the sum is passed to the activation function. Similarly to weights, the bias must be able to be either negative or positive for functional completeness. In the case of a molecular perceptron, a bias can be set by adding a set number of the participating ERN (for a negative bias), or ERN recognition site followed by the coding sequence for the output protein (for a positive bias). Because each ERN sequesters and degrades one RNA molecule, having a set number



(a) RecID: 02 (b) RecID: 03 (c) RecID: 04

(d) RecID: 05 (e) RecID: 06 (f) RecID: 07

(g) RecID: 08 (h) RecID: 09 (i) RecID: 10

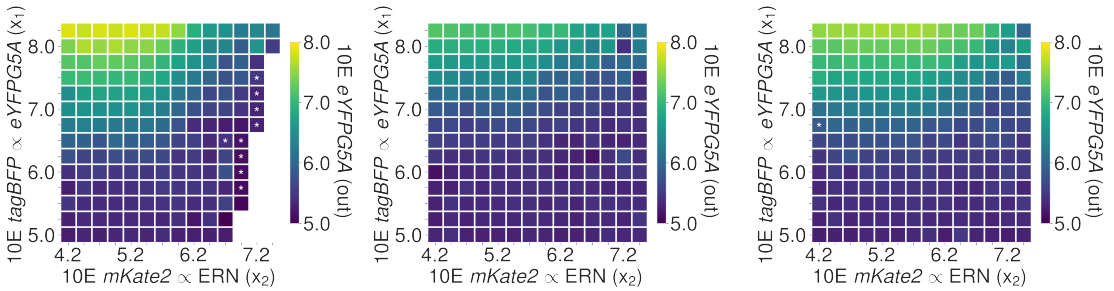
(j) RecID: WT (k) -ve ctrl

Figure 42: Compilation of all *CasE* recognition sites, as shown in Table 3. The negative control does not contain any *CasE* recognition sequence. If a pixel would contain less than 3 cells, it is excluded. Pixels that contain a white "*" are linearly interpolated. For analysis details see Section 5.6. Figure 42j is the response of the wildtype sequence, and serves as the positive control. Figure 42k does not contain a regulatory RNA sequence upstream of the start codon, and serves as the negative control. Plasmid pGW0010 abundance is shown on the x-axis of all subfigures. The plasmids transfected on the y-axis are identifiable by their RecID: 02: pGW0033. 03: pGW0034. 04: pGW0035. 05: pGW0036. 06: pGW0037. 07: pGW0038. 08: pGW0039. 09: pGW0040. 10: pGW0041. WT: pGW0042. -ve ctrl: pGW0043. Detailed plasmid maps for all plasmids are shown in Section 5.21.

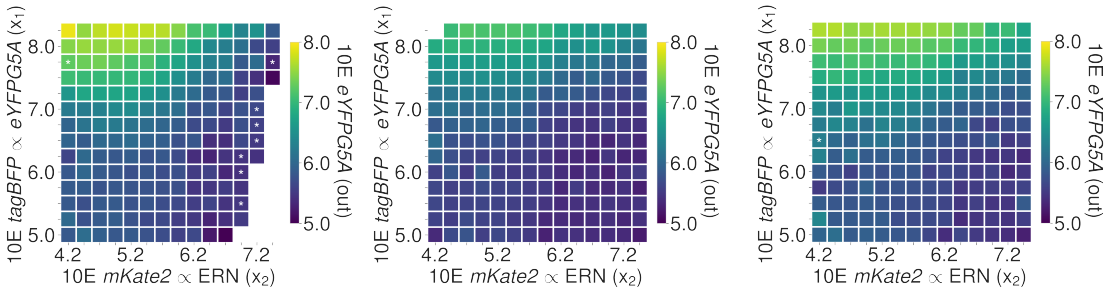
of ERNs in the system will act as a negative bias, since a baseline level of RNA for the positive output will already be degraded, i.e. subtracted from the system. The converse is true for the ERN recognition site followed by the coding sequence for the output protein for a positive bias, which sets a baseline of output from the bias alone, before any input (positive or negative) is added to the cell. For integrated circuits, the strength of the bias can be set via copy number, promoter strength, or Kozak sequence. Additionally, any other strategy that is outlined in Section 3.2 to tune weights can also be used to tune the strength of a bias. In this Section, I used poly-transfection to test the effects of a positive and negative molecular bias on ERN based perceptrons (*CasE*, *Csy4* and *pguCas13b*).

In the case of a positive bias, I transfected the gene construct expressing *eYFPG5A* with an upstream ERN recognition site with two different transfection markers: Firstly, *eYFPG5A* was correlated with *tagBFP*, and used as the input (x_1: *tagBFP* $\propto$ *eYFPG5A*). Separately, *eYFPG5A* was also correlated *iRFP720*, which could now be used as a positive bias (b: *iRFP720* $\propto$ *eYFPG5A*). Selecting a specific *iRFP720* essentially determined the level of background output, which could be regulated by the respective ERN. Figure 43 shows the effects of four levels of positive bias as measured by *iRFP720* $\propto$ *eYFPG5A* expression on three different ERN based perceptrons (*CasE*, *Csy4*, and *pguCas13b*). The respective ERNs were used as an input with a negative weight, and were measured via *mKate2* (x_2: *mKate2* $\propto$ ERN). As expected, the overall signal increased with the positive bias for each of the ERN based perceptrons. Note that the bias did not add a constant signal indiscriminately to each area of the perceptron. The top left area of each perceptron (where positive input is very high, and negative input is very low) remained at roughly the same level throughout, since the baseline (low bias) weighted sum of inputs was already enough to induce the maximum possible output after being passed to the activation function. Additionally, the output was still lower in the lower right area of each heatmap (where positive input is low, and negative input is high). This is because the high negative input was able to subtract enough from the positive bias to lead to a measurable decrease in final output.

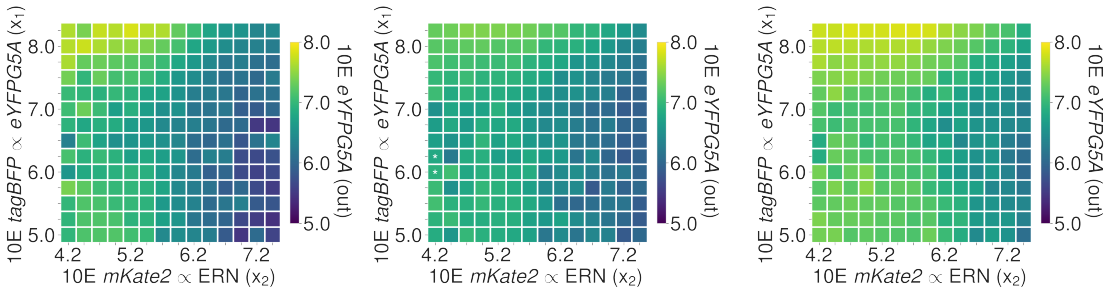
The bias bin width was set to 1 in log space (e.g. a bias of 6.0 has ten times less *iRFP720* signal than a bias of 7.0). I used the following plasmids for bias tuning: *CasE*: pGW0100. *Csy4*: pGW0101. *pguCas13b*: pGW0102. The ERNs are all shown on the x-axes and co-transfected with *mKate2* (pAK0006). *eYFPG5A* output plasmid abundances are shown on the y-axes, and co-transfected with *tagBFP* (pAK0022). The appropriate ERN recognition sites are encoded in output plasmids as follows: *CasE*: pGW0073. *Csy4*: pGW0103. *pguCas13b*: pGW0104. Detailed plasmid maps are shown in Section 5.21.



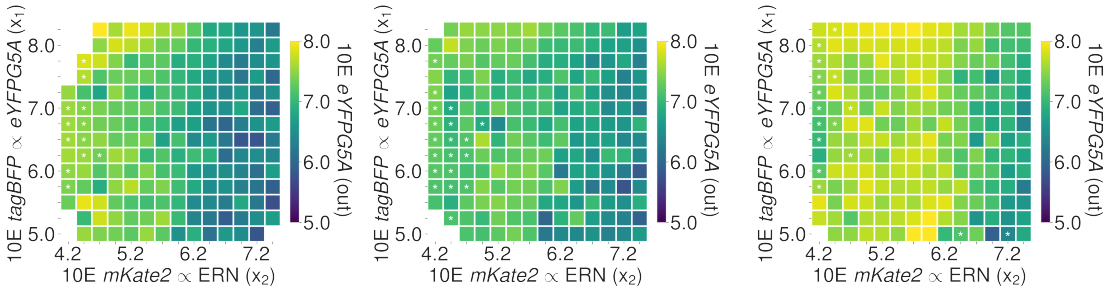
(a) ERN: *CasE*. Bias: 5.0 (b) ERN: *Csy4*. Bias: 5.0 (c) ERN: *pguCas13b*. Bias: 5.0



(d) ERN: *CasE*. Bias: 6.0 (e) ERN: *Csy4*. Bias: 6.0 (f) ERN: *pguCas13b*. Bias: 6.0



(g) ERN: *CasE*. Bias: 7.0 (h) ERN: *Csy4*. Bias: 7.0 (i) ERN: *pguCas13b*. Bias: 7.0



(j) ERN: *CasE*. Bias: 8.0 (k) ERN: *Csy4*. Bias: 8.0 (l) ERN: *pguCas13b*. Bias: 8.0

Figure 43: Results of three different molecular perceptrons at 4 different positive bias levels. Each column pertains to a specific ERN, and each row to a different bias level as measured by *iRFP* proxy (*iRFP* $\propto$ *eYFPG5A* (out)). *CasE* Figures: 43a, 43d, 43g and 43j. *Csy4* Figures: 43b, 43e, 43h, and 43k. *pguCas13b* Figures: 43c, 43f, 43i, and 43l. Low bias (*iRFP* = 5.0) Figures: 43a, 43b, and 43c. Medium bias (*iRFP* = 6.0) Figures: 43d, 43e, and 43f. High bias (*iRFP* = 7.0) Figures: 43g, 43h, and 43i. Very high bias (*iRFP* = 8.0) Figures: 43j, 43k, and 43l. Pixels with less than 3 cells are excluded. Pixels with a "*" have been linearly interpolated. For plasmid information see main text. For analysis details see Section 5.6.

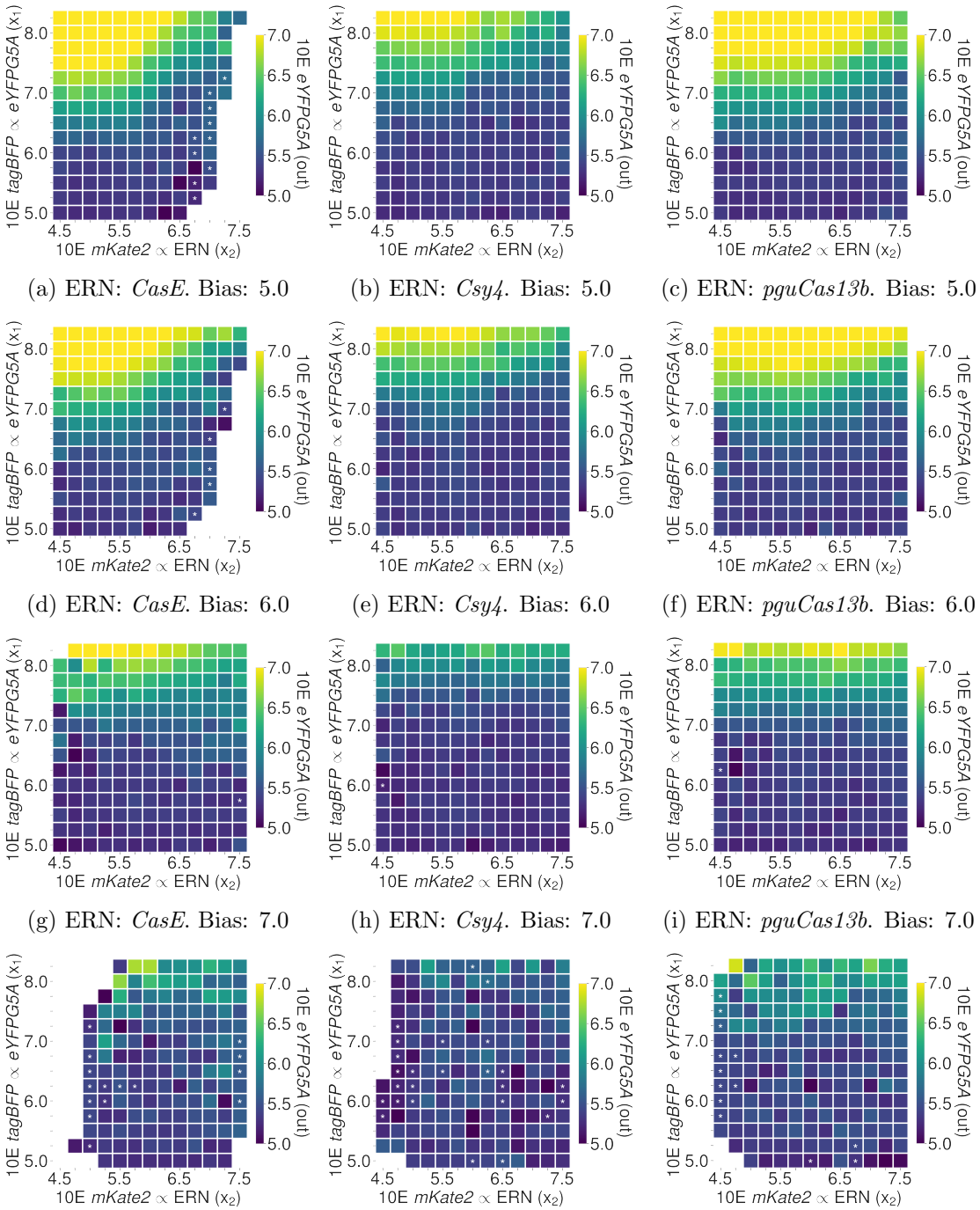
I used the same strategy to test the effects of a negative bias on a single molecular perceptron, with the change that *iRFP720* was correlated with an ERN (b: *iRFP720* $\propto$ ERN). I kept the inputs the same as for the experiment testing the effects of a positive bias (x_1: *tagBFP* $\propto$ *eYFPG5A*; x_2: *mKate2* $\propto$ ERN). Figure 44 shows the effects of four levels of negative bias, as measured by *iRFP720* $\propto$ ERN, for three different ERN based perceptrons (*CasE*, *Csy4*, and *pguCas13b*). As expected, the overall signal decreased with increasing negative bias for each of the ERN based perceptrons. As with the positive bias, adding the negative bias did not indiscriminately lower all output values by a set amount. If the original weighted sum (at low bias levels) returned the maximum possible value of the activation function, the value remained unaffected, until a high enough negative bias was added. This is especially apparent in the top left corner of each heatmap in Figure 44, between *iRFP720* bias levels 6.0 and 8.0. I used the same plasmids as for the positive bias tuning, which can be found in Section 5.21.

Adding positive and negative biases was the final piece needed to complete the functionality of a molecular perceptron. Now that this tuneability was confirmed, I could work on testing how individual molecular perceptrons would work together to form simple molecular neural networks.

3.4 Multilayer Molecular Perceptrons

As demonstrated in Sections 3.1, 3.2 and 3.3, ERNs can be used to construct molecular perceptrons with positive and negative inputs, tuneable positive and negative weights and both positive and negative biases. However, single perceptrons can only classify simple problems, depending on their activation function (e.g. only linear problems in two categories when based on a threshold function). This is in stark contrast to multi-layer perceptron networks (such as artificial neural networks), which are universal approximators for any arbitrary classification problem (Hornik, 1991; Murphy, 2013). In this Section, I connected multiple molecular perceptrons together to perform more complex functions. Firstly, I used two perceptrons in the same cell to perform a dual-region classification problem, using a single layer. Secondly, I connected three perceptrons to form a three-node molecular neural network to form a bandpass filter.

When manually designing molecular neural networks with multiple nodes, I found it intuitive to use "region algebra" to figure out how to connect the nodes. In region algebra, each shape can be added or subtracted from another shape. For instance, two triangles in opposite corners added together can form a dual region, where the middle down a diagonal line has low signal, and the two corners have high signal. Equivalently, each perceptron has



(a) ERN: *CasE*. Bias: 5.0 (b) ERN: *Csy4*. Bias: 5.0 (c) ERN: *pguCas13b*. Bias: 5.0

(d) ERN: *CasE*. Bias: 6.0 (e) ERN: *Csy4*. Bias: 6.0 (f) ERN: *pguCas13b*. Bias: 6.0

(g) ERN: *CasE*. Bias: 7.0 (h) ERN: *Csy4*. Bias: 7.0 (i) ERN: *pguCas13b*. Bias: 7.0

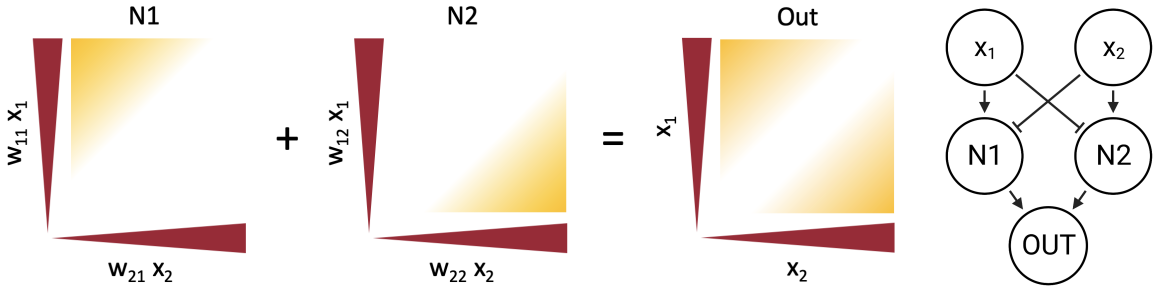
(j) ERN: *CasE*. Bias: 8.0 (k) ERN: *Csy4*. Bias: 8.0 (l) ERN: *pguCas13b*. Bias: 8.0

Figure 44: Results of three different molecular perceptrons (*CasE*, *Csy4*, and *pguCas13b*) at 4 different negative bias levels. Each column pertains to a specific ERN, and each row to a different bias level as measured by *iRFP* proxy ($iRFP \propto ERN$). *CasE* Figures: 44a, 44d, 44g and 44j. *Csy4* Figures: 44b, 44e, 44h, and 44k. *pguCas13b* Figures: 44c, 44f, 44i, and 44l. Low bias ($iRFP = 5.0$) Figures: 44a, 44b, and 44c. Medium bias ($iRFP = 6.0$) Figures: 44d, 44e, and 44f. High bias ($iRFP = 7.0$) Figures: 44g, 44h, and 44i. Very high bias ($iRFP = 8.0$) Figures: 44j, 44k, and 44l. Pixels with less than 3 cells are excluded. Pixels with a "*" have been linearly interpolated. For plasmid information see main text. For analysis details see Section 5.6.

a certain output, and is independent from the next perceptron, so each "shape" that is produced as an output can be added or subtracted to connected nodes. A simple example for this is shown in Figure 45, where two perceptrons are added together to form a dual region. The first perceptron, N1, is in the familiar format, where x_1 of the molecular perceptron is linked to a positive weight ($w_{11}>0$), whereas x_2 is linked to a negative weight ($w_{21}<0$). In the second perceptron, N2, the reverse is the case ($w_{12}<0$ and $2_{22}>0$). The first number in the weight index indicates which input it links from, whereas the second number indicates which node it is connected to (e.g. w_{12} links input x_1 to node N2). Note that the resulting dual region is distinct from the Boolean XOR gate in three main features: Firstly, the resulting output is still analogue, not simply a yes / no digital output. This can serve as a measure of confidence, that the classification is done correctly. Secondly, while an XOR gate would split the output layer into four squares, the decision boundary for this neuromorphic implementation are diagonal, illustrating the way signals are processed in an analogue way. Finally, a neuromorphic implementation allows for weights and biases to be applied to the network, resulting in different decision boundaries. For example, applying a negative bias to the output layer would result in the region without gene expression getting wider, while applying a positive bias would make it narrower. The XOR like design is interesting in another aspect: Previous work showed that a basic XOR gate is not possible with a single layer perceptron. Historically, the scientific community lost interest in researching artificial neural networks, because a hasty conclusion was inferred that multilayer perceptrons would also not be able to perform this operation (Minsky et al., 2017). Because of this history, the emulation of XOR-like functions via neural networks is still a prominent example to demonstrate the functionality of small neural networks. A network abstraction of the design for this dual region classifier is shown in Figure 45b.

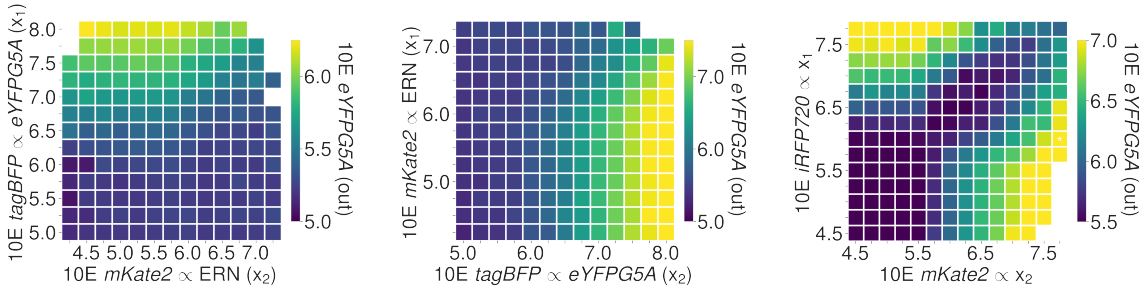
Figure 46 shows the results obtained via poly-transfection for the implemented network after a few rounds of manual weight adjusting. Figures 46a and 46b show the single perceptrons for the ERNs *Csy4* and *pguCas13b* respectively. Note that unlike most perceptrons presented in this thesis, Figure 46b displays the positive weight on the x-axis and the negative weight on the y-axis to match the region algebra example shown in Figure 45.

The molecular implementation for the complete dual region (shown in Figure 48b) is as follows: x_1 was measured by *iRFP720*, which was co-transfected in the ratio 1 *iRFP720* : 2 *pguCas13b* : 1 *Csy4rec_eYFPG5A*. This was to adjust the weight for the weaker *pguCas13b* ERN. x_2 was measured by *mKate2*, which was co-transfected in the ratio 1 *mKate2* : 1.5 *Csy4* : 1*pguCas13brec_eYGPG5A*. This adjustment was made to separate the two regions, and keep



(a) Region Algebra (b) Network

Figure 45: Example of how region algebra can be used to manually design a dual region classifier. Figure 45a shows how the outputs of two single perceptrons are added together, leading to high expression levels when either x_1 or x_2 are high, but not when both are high. This differs from a Boolean XOR gate, because the outputs are still analogue, not binary. Additionally, while an XOR gate would split the output layer into four squares, notably the decision boundary is diagonal. Figure 45b shows an abstraction of the resulting network.



(a) *Csy4* Perceptron (b) *pguCas13b* Perceptron (c) Dual Region

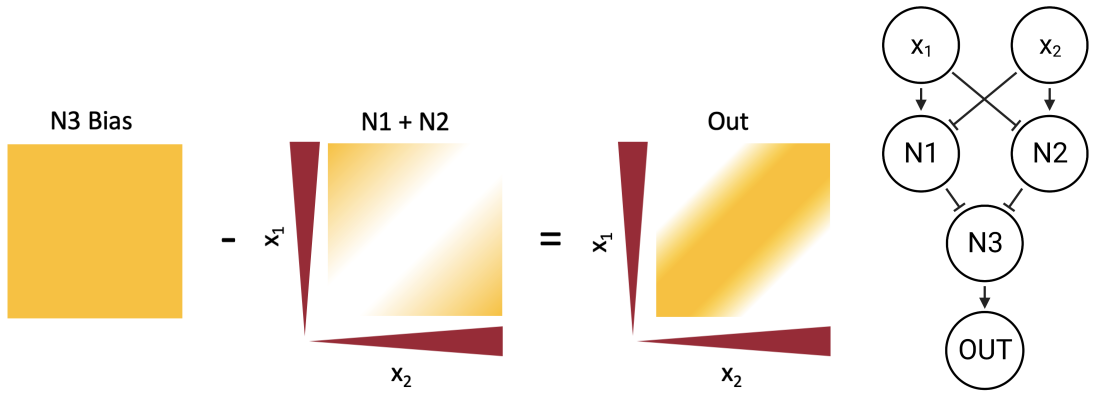
Figure 46: Results of a manually designed dual region classifier, following the design obtained via region alegbra shown in Figure 45. Figures 46a and 46b show single perceptrons based on the ERNs *Csy4* and *pguCas13b* respectively. Figure 48b shows the dual region resulting from the two perceptrons being added together. For plasmid information and tuned weight information see main text.

decision boundaries parallel. The exact plasmid maps for these can be found in 5.21, and are as follows: *iRFP720*: pAK0024. *mKate2*: pAK0006. *CasE*: pGW0100. *Csy4*: pGW0101. *pguCas13b*: pGW0102. *Csy4rec_eYFPG5A*: pGW0103. *pguCas13brec_eYFPG5A*: pGW0104.

As seen in Figure 46c, the two regions of the dual region classifier are distinctly separated by a region of low output expression. Additionally, the decision boundary displays the above described characteristics of a neuromorphic implementation: The output is still analogue, the decision boundary is diagonal and graded, and the weaker repression via the less strong ERN *pguCas13b* is mitigated by tuning the weights of node N2.

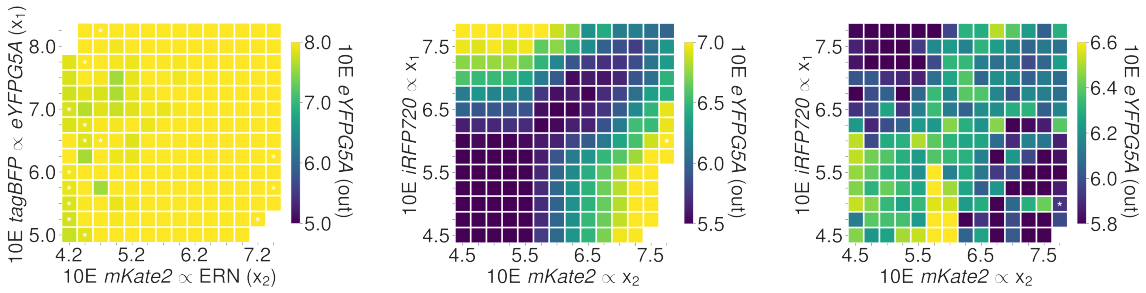
Following the same intuitive design strategy of region algebra, I also designed a bandpass classifier. This is shown in Figure 47, where a dual region (which is a sum of two nodes in a hidden layer) is subtracted from a high bias in a third node. The hidden layer in this case

consists of the two nodes N1 and N2 (as seen in Figure 47b), because no direct measurement of the two nodes can be taken. Instead, both produce an ERN, which acts as a negatively weighted input for a third node with a positive bias. While still possible, not being able to measure the hidden layer directly via fluorescent proteins made the weight optimisation process a lot harder.



(a) Region Algebra (b) Network

Figure 47: Example of how region algebra can be used to manually design a bandpass classifier; the inverse of the dual region shown in Figure 45. Figure 47a shows how a dual region classifier is subtracted from a high bias, leading to low output levels when either x_1 or x_2 are high, and high output in other cases. Figure 47b shows an abstraction of the resulting network.



(a) Bias (b) Dual Region (c) Bandpass

Figure 48: Results of a manually designed bandpass classifer, following the design obtained via region algebra shown in Figure 47. Figure 48a shows an example of a high positive bias. Figure 48b shows the dual region, which is a sum of nodes N1 and N2 as shown in Figure 46. Figure 48c shows the bandpass resulting from subtracting a dual region from a randomised bias. Node N1 is based on *Csy4*, N2 on *pguCas13b* and N3 on *CasE*. For plasmid information and weight details see main text.

Figure 48 shows the molecular implementation for the bandpass network designed following the region algebra in Figure 47. N1 and N2 are mapped similarly to the dual region case, but output the ERN *CasE* instead of a fluorescent protein. In the third node, N3, a randomised bias was applied to all cells, which can be repressed by *CasE*. x_1 was measured by

iRFP720, which was co-transfected in the ratio 1 *iRFP720* : 2 *pguCas13b* : 1 *Csy4rec_CasE*. This was to adjust the weight for the weaker *pguCas13b* ERN. x_2 was measured by *mKate2*, which was co-transfected in the ratio 1 *mKate2* : 1 *Csy4* : 1 *pguCas13brec_CasE*. The randomised bias consists of 1 *tagBFP* : 1 *CasErec_eYFPG5A*. The exact plasmid maps for these can be found in 5.21, and are as follows: *iRFP720*: pAK0024. *mKate2*: pAK0006. *tagBFP*: pAK0022. *CasE*: pGW0100. *Csy4*: pGW0101. *pguCas13b*: pGW0102. *Csy4rec_CasE*: pGW0116. *pguCas13brec_CasE*: pGW0117. *CasErec_eYFPG5A*: pGW0073.

As seen in Figure 48c, the top left and bottom right corners (where either of the inputs is high, but not both), both show low levels of output. The two regions are separated by a region of higher gene expression, resembling the manually designed bandpass shown in Figure 47. However, in comparison to the dual region classifier, the data is more noisy, and the levels of high gene expression are not as distinctly regulated, indicating a lower degree of confidence in classification. The noise can be attributed to three main reasons. Firstly, the addition of a random bias, which was necessary because of the limitations with a four colour poly-transfection experiment with the utilised flow cytometer. Secondly, because with each additional hidden layer, the noise increases, if no clean-up layer is added. This is because noise from one layer gets carried forward, leading to a lower degree of confidence in classification. And thirdly, the inability to pick optimal weights. The optimisation process already took several manual iterations in the lab, costing time and resources. Furthermore, manual weight adjustment is likely not going to yield the optimal result. This illustrates that purely manual design and weight tuning can only go so far. To move towards general neuromorphic computing, a better weight optimisation strategy needs to be developed.

3.5 Updating Molecular Perceptron Weights *in vivo*

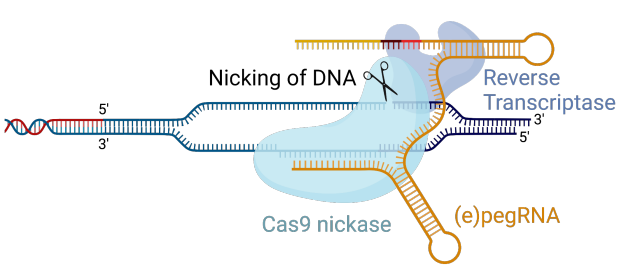
3.5.1 Introduction to prime editing

As illustrated in Section 3.4, manually designing molecular neural networks is possible, but held back by tedious iterations of manually adjusting the weights and biases of each node in order to build the correct classifier. This greatly reduces the advantage that neural networks usually have over logic gate computation, which is that the same network layout can be used for different classification problems, by simply adjusting its weights and biases. In biology, I think this problem can be tackled from two different angles: Firstly, by creating a predictive model that understands how molecular perceptrons work and interact, weights and biases can be trained *in silico* on biological data, and then compiled into DNA. The DNA can then be transfected into cells, with the network already tuned for the task at hand.

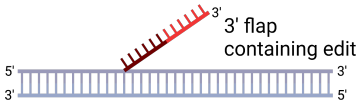
However, if the predictive model is lacking in any area, this will only put the network in roughly the right ballpark, and weights will still need to be tuned and adjusted manually.

Another strategy is to train the weights already in a biological context, for example via reinforcement learning (Racovita et al., 2022). In order to enable an *in vivo* strategy in mammalian cells, it must be compatible with genome integration. This means storing weights as multiple copies of the same gene can become problematic, since precise weight adjustment becomes difficult once multiple copies of the same gene exist. Additionally, adjusting a weight must not prohibit the weight from being adjusted again - otherwise only a single improvement step can be taken, greatly limiting weight optimisation strategies. Furthermore, the user must be able to direct whether the weight increases or decreases, and the weight must remain stable unless the user provides an input. This allows for supervised learning, where the user provides learning feedback for the cells. Finally, there must be a way to confirm that weight adjustment has occurred. While invaluable for prototyping, this also allows for cells where weight adjusting did not occur to be discarded, e.g. via fluorescence assisted cell sorting (FACS). One way to implement a strategy that conforms to all these restrictions is via prime editing, which allows for extensive gene editing downstream of an `NGG` protoscpacer adjacent motif (PAM) sequence (Anzalone et al., 2019).

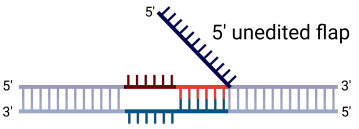
The prime editing machinery consists of three parts that were optimised via directed evolution: A *Cas9* nickase, a reverse transcriptase protein, and an (engineered) prime editing guide RNA ((e)pegRNA) (Anzalone et al., 2019; Nelson et al., 2022). A schematic representation of the prime editing process is shown in Figure 49. The *Cas9* nickase and the reverse transcriptase are fused via a flexible linker, and *Cas9* binds the (e)pegRNA; thus all three parts of the machinery are co-localised. The (e)pegRNA consists of separate functional regions: The first part is a protospacer. The protospacer targets the prime editing machinery to a specific DNA sequence via *Cas9* and Watson-Crick base pairing, leading to the complex shown in Figure 49a. The nick occurs between bases 3 and 4 upstream of the PAM sequence (Jinek et al., 2012; Anzalone et al., 2019). While the protospacer RNA sequence is bound to the 3' strand of the targeted DNA, the 5' strand is unpaired, forming an open loop, keeping the bases on the 5' strand exposed. A primer binding site on the (e)pegRNA is then able to bind these exposed bases, forming an RNA-DNA duplex that can be used to prime the reverse transcriptase, which is co-localised to the open loop structure by *Cas9*. This structure is shown in Figure 49b. The primed reverse transcriptase then extends the nicked 5' strand, based on the sequence encoded in the (e)pegRNA. Since the *Cas9* part of the prime editing machinery nicks the 5' strand, the cell repairs the nick based on the nicked RNA-DNA duplex



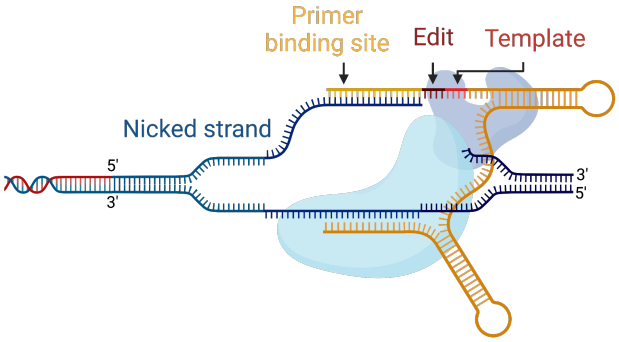
(a) *Cas9* nickase complex



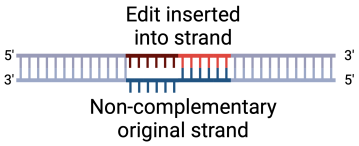
(c) Unincorporated edit



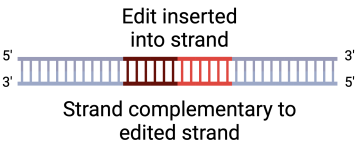
(d) Edit incorporation (nicked)



(b) (e)pegRNA primer binding



(e) Nick repair



(f) Final product

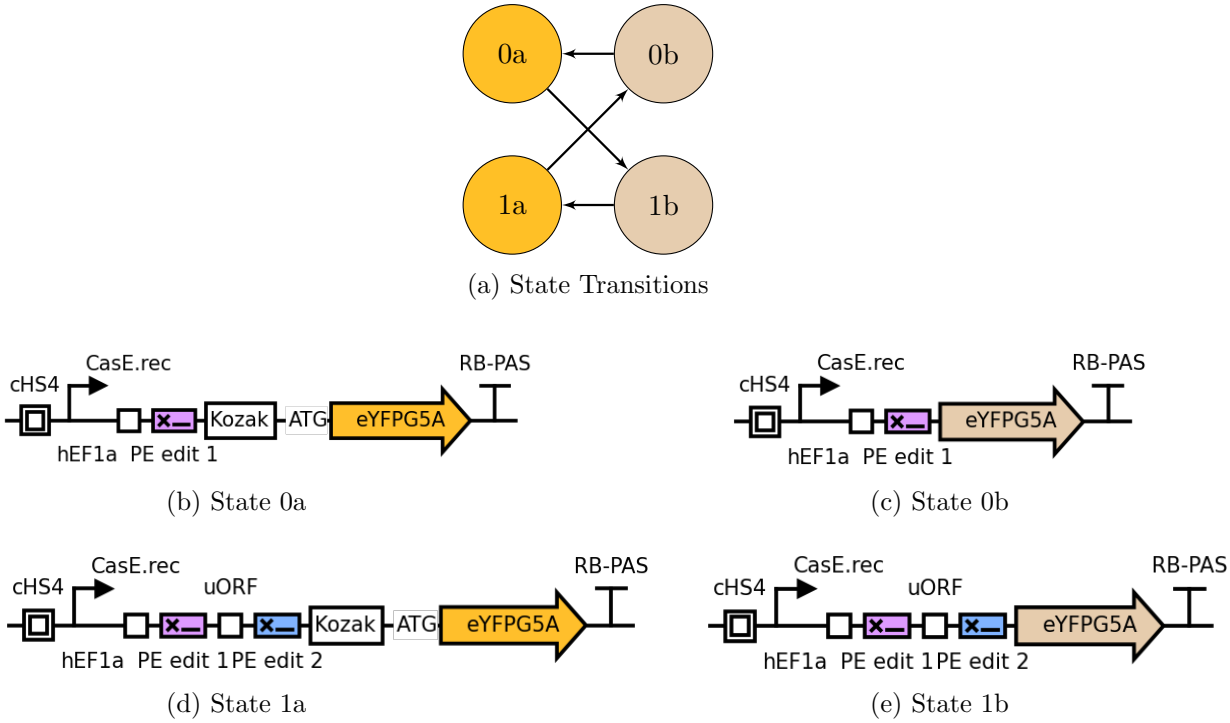
Figure 49: Schematic representation of the prime editing process as described by Anzalone et al. (2019). The protein machinery consists of a *Cas9* nickase domain, which is fused to a reverse transcriptase. The protein machinery is then bound to an (e)pegRNA. Figure 49a shows how the machinery is bound to a DNA strand and introduces a nick in a target 5' strand. The nicked strand can then be bound by a primer binding site in the (e)pegRNA (Figure 49b). The reverse transcriptase then extends the nicked strand from the 5' to 3' direction, transcribing the edit, as well as a template region, encoded in the (e)pegRNA. This can lead to two possible outcomes. Firstly, an extended 3' flap (Figure 49c), which is not incorporated during nick repair. The second possibility is that the template strand binds the target DNA, leading to a nicked DNA with the 5' unedited flap unbound to the complementary DNA strand (Figure 49d). This flap is then cleared during nick repair (Figure 49e). This leads to a region containing the edit being mismatched. After mismatch repair, the desired edit is incorporated and stable (Figure 49f).

that was created by the reverse transcriptase. Two DNA strands now exist in the 5' direction: One created by the reverse transcriptase via RNA priming (Figure 49c), and the original, nicked, 5' strand that is replaced (Figure 49d). The nick repair is possible due to a template region (bright red in Figure 49) that is kept identical to the original sequence, which allows the extended 5' flap to bind the original DNA (Figure 49e). This leads to a mismatched DNA strand, which is repaired into the desired DNA sequence (Figure 49f). The repaired sequence now includes the desired edit, downstream of the sequence bound by the primer binding site and the protospacer. While not essential, it is beneficial to delete the PAM site in this encoded DNA. This allows the prime editing machinery to only bind unedited DNA, biasing the reaction towards the desired edit. A secondary nick (ideally around 40-90 bp) upstream of the edit on the 3' strand can also enhance repair efficiency (by an average of 41%) (Anzalone et al., 2019; Scholefield and Harrison, 2021; Nelson et al., 2022).

3.5.2 Design of a Weight Updating Algorithm

In this section, I propose a strategy to tune the weights of a molecular perceptron using prime editing (Anzalone et al., 2019). A schematic overview of the weight tuning strategy is shown in Figure 50. In essence, the developed strategy allows for repeated uORF insertion (and theoretically deletion) upstream of a gene of interest in order to tune the weight of that particular gene down (or up), as demonstrated in Section 3.2. I specifically designed this strategy for single copy integration of the circuit (e.g. via a landing-pad, or twin prime editing strategy (Gaidukov et al., 2018; Anzalone et al., 2021)). Importantly, the molecular algorithm I developed to adjust weights *in vivo* can also be applied multiple times, which is a prerequisite for learning strategies. Additionally, each editing step is distinct, and can be measured via flow cytometry.

These properties are enabled via a two step editing process, proof of concept of which is demonstrated for a *CasE* based molecular perceptron, where the positive weight (w_1) is tuned on *eYFPG5A*. A schematic of all states needed to have full tuneability, where increasing as well as decreasing the weight is possible, is shown in Figure 50. Overall, the states are labelled based on the number of uORFs upstream of the coding sequence, and their ability to express protein, where states ending in "a" can express protein, whereas states ending in "b" cannot. For example, state 1b would include one uORF, and is unable to express protein. The initial state (0a) is shown in Figure 50b, which shows a regular gene expression cassette for *eYFPG5A*, with an upstream *CasE* recognition site. Note that a purple prime editing site serves as one of the homologous target sequences for repair (analogously to the bright red sequence shown in Figure 49). The initial state (which is equivalent to a maximum weight of 1)



(a) State Transitions

(b) State 0a

(c) State 0b

(d) State 1a

(e) State 1b

Figure 50: Minimal states to demonstrate full tuneability of weights of a molecular perceptron via prime editing. The states are labelled based on the number of uORFs upstream of the coding sequence, and their ability to express protein (state Xa can express protein, state Xb cannot, where X is the number of uORFs). The possible state transitions are shown in Figure 50a. State 0a (Figure 50b) is a regular gene expression cassette to express *eYFPG5A*, with an upstream *CasE* recognition site, and an upstream prime editing (PE) target sequence (purple). Using prime editing the sequence between the PE edit 1 sequence and the beginning of the gene can be changed to include a uORF, and disable gene expression, resulting in state 1b (Figure 50e). This inactive state can be restored via prime editing by changing the sequence between the blue PE edit 2 sequence and the gene, resulting in state 1a (Figure 50d). State 1a can then be used to either insert another uORF (not shown), or delete one, resulting in state 0b (Figure 50c). State 0b can then be restored to state 0a. All states are also assembled. Plasmid names are shown in Table 4.

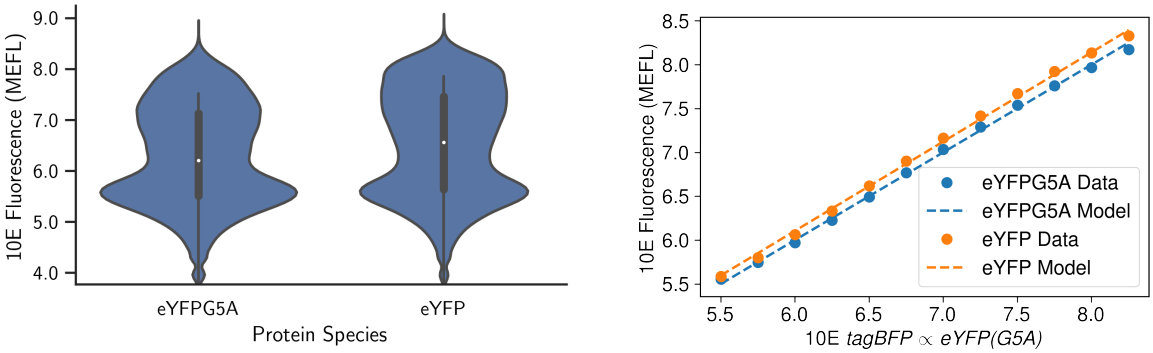
can be tuned down, by inserting a regulatory sequence, in this case a uORF. Additionally, the Kozak sequence, as well as the `ATG` start codon and an additional base is deleted, introducing a frameshift to *eYFPG5A*. Importantly, this deletion makes the resulting state 0b (Figure 50c) screenable via flow cytometry, allowing for fluorescence assisted cell sorting to exclude unedited cells. Note that this deletion also includes the PAM sequence, which is present at the beginning of all synthetic eukaryotic coding sequences due to the last three base pairs (`ACC`) of the Kozak sequence. Importantly, I deigned the entire prime editing machinery to bind in the reverse direction, since the uORF is inserted upstream of the primer binding site. This allows the *Cas9* nickase to bind the `NGG` PAM sequence on the 3' strand in the Kozak sequence.

Apart from the inserted uORF, a second distinct prime editing site (coloured blue) is also introduced to serve as a homology region for the next editing step. All even states (0a,b; 2a,b; ...) have the same edit site (coloured in purple) proximal to the gene to be tuned. The same is true for the uneven states, which also have their own distinct prime editing site (coloured blue). In the second step (changing from state Xb to Xa), expression is restored by reintroducing the Kozak and `ATG` sequences, and fixing the frameshift mutation. This is done based on the homology site specific to the state to be repaired, and a secondary PAM sequence somewhere in the beginning of the gene of interest. Since this edit results in restored gene expression, switching from any state Xb to Xa can also be detected via flow cytometry, enabling cell sorting.

Note that being able to transition between all of those states results in repeatable tuning, where every possible weight via uORF insertion should be reachable: Following Figure 50, state 0a (Figure 50b) becomes state 1b (Figure 50e) by introducing a uORF. Restoring gene expression to state 1b results in state 1a (Figure 50d). State 1a can then either be used as a starting point to add another uORF, resulting in state 2b (not shown), or the uORF can be deleted, resulting in state 0b (Figure 50c). Again, restoring gene expression leads to state 0a, closing the editing loop. Note that if the starting point is state 0a, theoretically only uORF insertion, not uORF deletion is necessary to reach every possible weight by uORF tuning.

3.5.3 Implementation of a Weight Updating Algorithm

The prerequisite that needs to be met to implement the strategy discussed in Section 3.5.2 is that two PAM sequences need to be present (one for each editing step). As discussed, the first one is already present due to the Kozak sequence. The second PAM sequence will differ on a case by case basis, since it follows after the Kozak sequence. In case of the designed example, where the weight of a positive output is tuned, I needed to find a second PAM



(a) Violin plot

(b) Line plot

Figure 51: Comparison of *eYFP* and *eYFPG5A* to confirm that the amino acid substitution maintains protein function. Figure 51a shows the fluorescent output in MEFL for both protein species in a violin plot. The centre bar shows a box plot. Figure 51b shows a line fit to a 1:1 co-transfection correlating *eYFP* or *eYFPG5A* with *tagBFP*. As seen, *eYFP* and *eYFPG5A* have roughly the same fluorescent output. The intercept is set to 0. The optimal slopes are roughly 1.02 for *eYFP* ($R^2 \approx 0.99$), and 1.00 for *eYFPG5A* ($R^2 \approx 0.99$). Plasmids pGW0071 and pGW0020 are used to express *eYFPG5A* and *eYFP* respectively. Both are co-transfected with *tagBFP* (plasmid pAK0022). Detailed plasmid maps are shown in Section 71.

sequence on the reverse strand in the first few bases of *eYFP*. A quick inspection revealed that the first natural usable PAM sequence is encoded >25 bases downstream of the edit (in the Threonine codon `ACC`), which is too far to ensure good editing efficiency. I therefore created a genetic *eYFP* variant, where I substituted Glycine at position 5 (codon `GGG`) with Alanine (codon `GCC`). To ensure that this mutation does not impede the function of *eYFP*, I compared the outputs of the two fluorescent proteins. As seen in Figure 51, *eYFPG5A* still maintains close to identical fluorescent properties. Figure 51a shows the distributions of transfections with *eYFPG5A* and *eYFP*. Figure 51b shows two co-transfections, where *eYFP* and *eYFPG5A* are correlated with another fluorescent protein (*tagBFP*) in a 1:1 ratio. I then fit a line to both co-transfections, where both lines go through the origin. As shown, both fluorescent values increase linearly. *eYFP* has a slope of ca. 1.02 ($R^2 \approx 0.99$), and *eYFPG5A* has a slope of ca. 1.00 ($R^2 \approx 0.99$); the difference between the two may well be due to manual error. I calibrated all fluorescent values based on standard beads, and converted them to MEFL, as described in Section 5.6. I used *eYFP* (not *eYFPG5A*) as the standard colour for MEFL colour conversion.

This shows that even in this non-optimal scenario, where no natural PAM sequence is present, and no silent mutation is possible, a simple amino acid substitution can enable this weight tuning strategy. (In the case of *eYFP*, a silent mutation would have been possible by changing codon for Serine on position 3 from `AGC` to `TCC`.) Alternatively, a neutral amino

Function	Name
State 0a	pGW0077
State 0b	pGW0078
State 1a	pGW0079
State 1b	pGW0080
epegRNA: 0a → 1b	pGW0060
epegRNA: 0b → 0a	pGW0070
epegRNA: 1a → 0b	pGW0067
epegRNA: 1b → 1a	pGW0069
epegRNA: Secondary Cut	pGW0059
iRFP720	pAK0024
tagBFP	pAK0022

Table 4: Plasmid names for the pU6 epegRNA expression plasmids, and for the states shown in Figure 50. Detailed plasmid maps can be found in Section 5.21.

acid like Alanine could be included after the start codon. Note that, since the start codon should be deleted, the PAM sequence cannot be included between the Kozak sequence and the start codon.

As illustrated in Section 3.5.2, full tuneability of biological weights is possible if three editing steps can be implemented: uORF insertion, expression repair, and uORF deletion. As discussed, each step in the editing process should be screenable via flow cytometry. To check whether an edit has occurred, I compared the expression of the edited expression cassette to a negative control. Each experiment (edited and negative control) consisted of a poly-transfection, where the prime editing machinery was correlated with one transfection marker (*iRFP720*), and the plasmid to be edited was correlated with a different transfection marker (*tagBFP*). For the editing experiment, the prime editing machinery included the *PEmax* enzyme, an epegRNA containing the edit, and an epegRNA to introduce the secondary nick upstream of the edit site. The ratio was set to 1 *iRFP720* : 3 *PEmax* : 1 epegRNA for the edit : 1 epegRNA for the secondary nick (Nelson et al., 2022). The negative control was identical, but was missing the epegRNA that includes the edit. Instead, it contained twice the amount of epegRNA for the secondary nick, to keep the total amount of DNA identical. I then used the fold change of the editing experiment over the negative control to infer whether or not the edit was successful. Plasmid names for each state and epegRNA are shown in Table 4. Detailed plasmid maps are shown in Section 5.21.

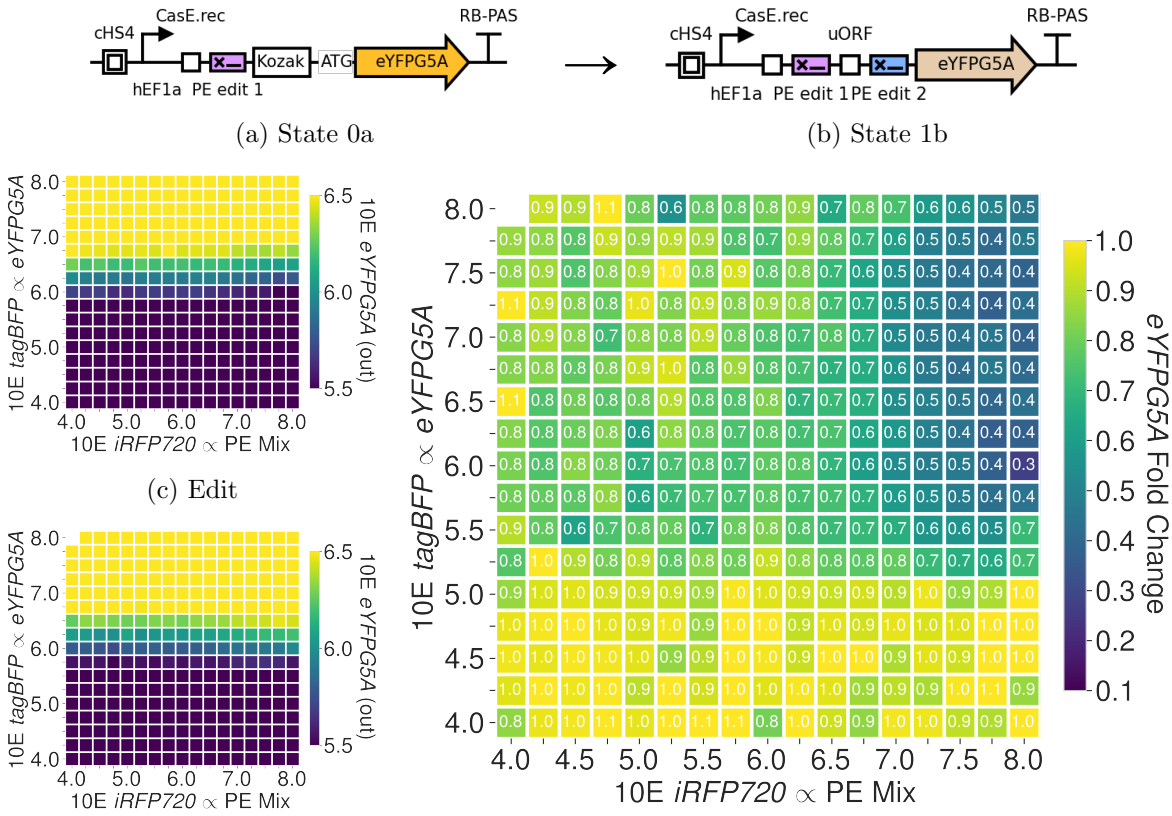
When starting at state 0a, the first step is to insert a uORF sequence (going from the plasmid shown in Figure 50b to the plasmid in Figure 50e). Since this change impedes gene expression, a reduction of gene expression is expected when compared to the negative control. This is shown in Figure 52. As can be seen in Figure 52e, gene expression was indeed reduced by up to 70% at high levels of the prime editing machinery, as soon as a threshold of the state

0a plasmid was reached. Note that this decrease in gene expression was only immediately apparent when looking at the fold change. For a state Xa to Xb change, I expected this behaviour during a transient transfection because of two main reasons: Firstly, multiple copies of the expression plasmid are present in each cell, if a transfection is successful. Since editing efficiency is not at 100%, *eYFPG5A* expression is still to be expected. This can be mitigated by integrating a single copy of the expression plasmid for *eYFPG5A*. Secondly, even assuming 100% editing efficiency, *eYFPG5A* still needs time to degrade for the signal to go down. Since I transfected both the editing machinery and the *eYFPG5A* expression plasmid at the same time, *eYFPG5A* is present when the editing machinery becomes active. Transfecting the editing machinery before the *eYFPG5A* expression plasmid may mitigate this problem. Single-copy integration (which is the final implementation goal for this weight updating algorithm) has the additional benefit, that more time can pass before fluorescence is measured, allowing for more *eYFPG5A* degradation. Additionally, a degradation tag may be added to *eYFPG5A* to further increase the difference between the editing experiment (Figure 52c) and the negative control (Figure 52d).

The second step in the weight adjustment algorithm I designed is to restore expression capabilities after the Kozak and `ATG` sequences are deleted, and a frameshift is introduced (either during uORF insertion or deletion). To test this, I screened the editing step from state 0b (Figure 50c) to state 0a (Figure 50b). I assumed that this expression restoration step is the easiest to detect, since no uORF sequence is present, leading to maximal gene expression per corrected plasmid. The restoration of gene expression (changing from state Xb to Xa) is also much easier to screen (compared to a change from state Xa to Xb), since without an edit, no baseline fluorescence should be produced. Therefore this editing step was not complicated by the issues of protein degradation dynamics, such as with uORF insertion or deletion.

Figure 53 shows the restored *eYFPG5A* expression. As seen in Figure 53c, a clear signal appeared at high levels of the editing machinery and the inactive *eYFPG5A* plasmid. Comparing the signal with the negative control (Figure 53d) resulted in Figure 53e. This shows that the signal increased up to ca. 15 fold when compared to the negative control, confirming the desired edit.

For a complete editing cycle, adjusting the weights between all the states shown in Figure 50 must be achieved. Two state changes are still needed to complete the editing cycle: The transition from state 1b to state 1a, and the transition from state 1a to 0b. Both of these steps are more difficult to screen than the previous two editing steps, since the fluorescent



(a) State 0a

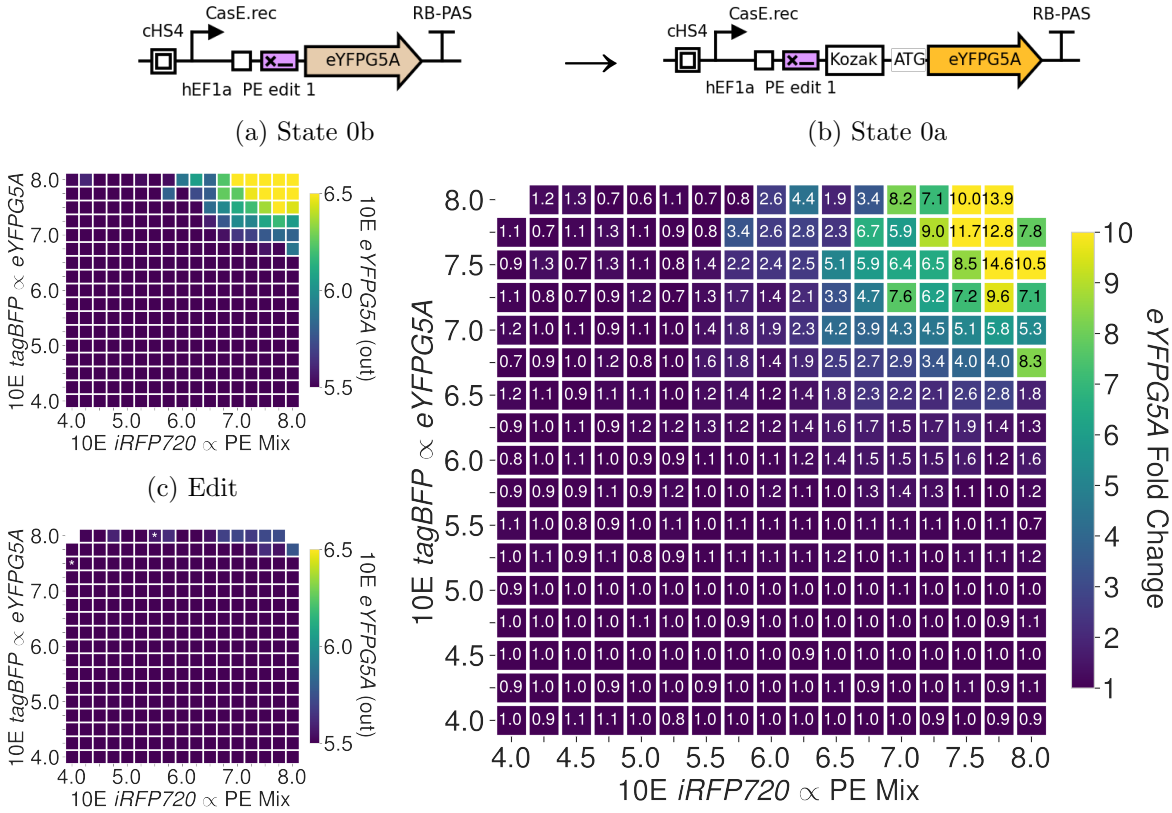
(b) State 1b

(c) Edit

(d) -ve control

(e) Fold change

Figure 52: Insertion of a uORF sequence upstream of *eYFPG5A*, where gene expression is also turned off, switching from state 0a (Figure 52a) to state 1b (Figure 52b). Other states are shown in Figure 50. The abundance of prime editing machinery is shown on the x-axis, the abundance of the *eYFPG5A* expression plasmid on the y-axis. The colourbar indicates the measured fluorescence of *eYFPG5A*. All units are in MEFL; measurements are taken 3 days after transfection. Figure 52c shows the results with complete editing machinery present. Figure 52d shows the negative control, where the editing epegRNA is missing. Figure 52e shows the fold change of *eYFPG5A* fluorescence of the edit over the negative control. As expected, the relative fluorescence decreases with increasing concentration of the editing machinery, once a certain threshold of *eYFPG5A* expression is reached. Flow cytometry data for this dataset were collected by Charles van de Mark.



(a) State 0b

(b) State 0a

(c) Edit

(d) -ve control

(e) Fold change

Figure 53: Restoration of Kozak and `ATG` sequences and frameshift repair, switching from state 0b (Figure 53a) to state 0a (Figure 53b). Other states are shown in Figure 50. The abundance of prime editing machinery is shown on the x-axis, the abundance of the (defunct) *eYFPG5A* expression plasmid on the y-axis. The colourbar indicates the measured fluorescence of *eYFPG5A*. All units are in MEFL; measurements are taken 3 days after transfection. Figure 53c shows the results with complete editing machinery present. Figure 53d shows the negative control, where the editing epegRNA is missing. Figure 53e shows the fold change of *eYFPG5A* fluorescence of the edit over the negative control. As expected, the relative fluorescence increases significantly at high concentrations of editing machinery and *eYFPG5A* plasmid. Flow cytometry data for this dataset were collected by Charles van de Mark.

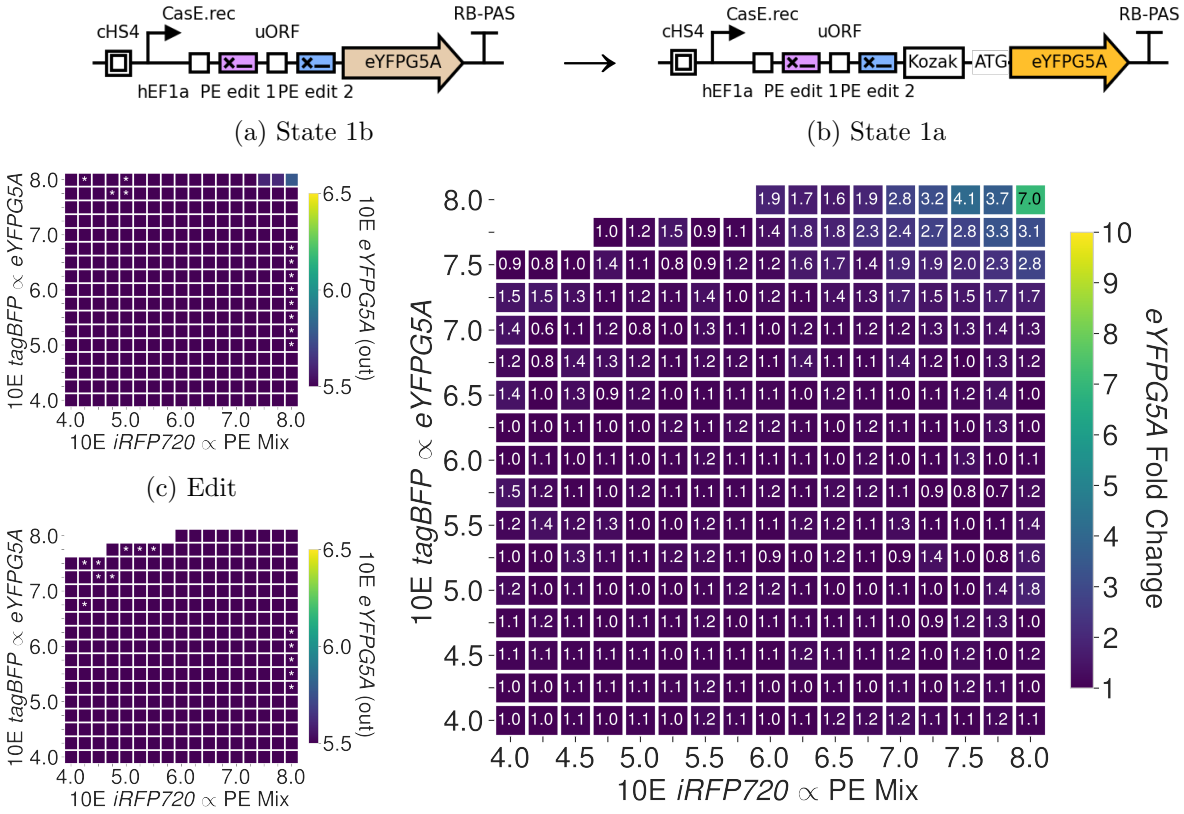
state (1a) includes a uORF, which results in lower signal difference after a correct editing step.

The transition from state 1b to state 1a follows the uORF insertion step shown in Figure 52, and should be easier to screen between the two remaining editing steps, since a fluorescent protein is being expressed. Therefore, after a correct edit, fluorescence should increase, although to a lesser extent than when no uORF is present. Indeed Figure 54 shows how fluorescence increased up to 7 fold when compared to the negative control, indicating editing was successful. This fold change is about half of the change observed when no uORF was present as gene expression was restored.

The final transition, state 1a to 0b, is needed to revert any uORF insertions that were performed in previous edits. This is useful, in case the weights are tuned down too much during weight optimisation. Since a uORF is present in the initial state (state 1a), the signal strength of a single DNA is already diminished compared to state 0a. This means a successful edit is not as visible as in the state 0a to state 1b transition, and therefore a lesser drop in signal strength is expected. This is indeed observed in Figure 55, where successful edits are observed, and lead to a 40% reduction in gene expression when compared to the control, compared to the 70% reduction observed in the state 0a to state 1b transition (Figure 52. This edit was the final step needed to freely add and remove uORFs in the context of a molecular perceptron, marking a first step to enable algorithmic weight tuning strategies such as *in vivo* backpropagation or supervised learning in mammalian cells.

3.6 Discussion & Conclusion: Neuromorphic Computing in Synthetic Biology

Artificial neural networks are currently the most utilised pattern recognition algorithm used in computer science (Srivastava et al., 2021). One of the main advantages of artificial neural networks is that the same structure can be used for different classification tasks, by simply changing the weights and biases of the individual perceptrons. This advantage alone could be vastly beneficial to synthetic biology, where the current state of the art for information processing is based on Boolean logic (Nielsen et al., 2016; Espah Borujeni et al., 2020). Neuromorphic computers are primed for neural network applications, since they process analogue signals and can inherently store weights and biases. Adding neuromorphic components to synthetic biology could greatly improve the design, build, test and learn cycle of synthetic biology, leading to accelerated research and translation e.g. in cell therapies (Zhao et al., 2023; Kitada et al., 2018; Meng and Ellis, 2020).



(a) State 1b

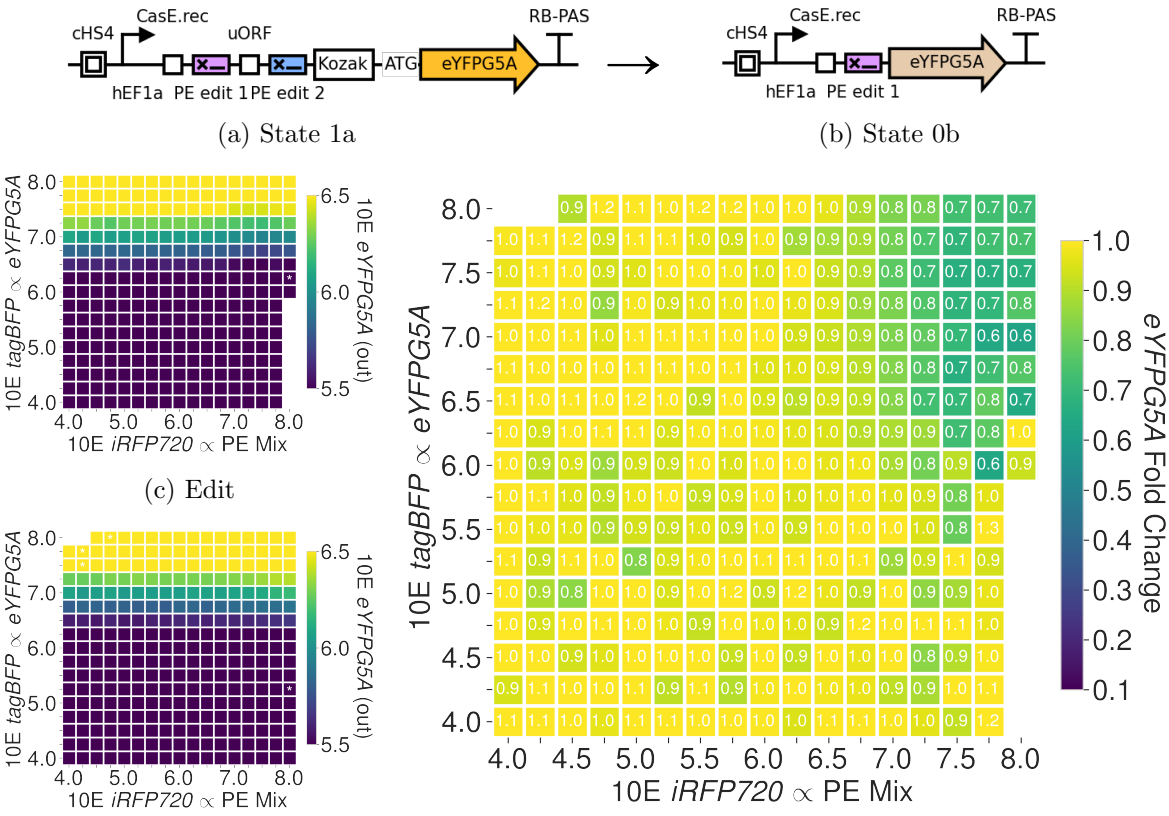
(b) State 1a

(c) Edit

(d) -ve control

(e) Fold change

Figure 54: Restoration of Kozak and `ATG` sequences and frameshift repair, switching from state 1b (Figure 54a) to state 1a (Figure 54b). Other states are shown in Figure 50. The abundance of prime editing machinery is shown on the x-axis, the abundance of the (defunct) *eYFPG5A* expression plasmid on the y-axis. The colourbar indicates the measured fluorescence of *eYFPG5A*. All units are in MEFL; measurements are taken 3 days after transfection. Figure 54c shows the results with complete editing machinery present. Figure 54d shows the negative control, where the editing epegRNA is missing. Figure 54e shows the fold change of *eYFPG5A* fluorescence of the edit over the negative control. As expected, the relative fluorescence increases at high concentrations of editing machinery and *eYFPG5A* plasmid. As predicted, the fold change is less than the fold change for Figure 53, where no uORF is present. Transfection and flow cytometry for this dataset were performed by Charles van de Mark.



(a) State 1a

(b) State 0b

(c) Edit

(d) -ve control

(e) Fold change

Figure 55: Deletion of a uORF sequence upstream of *eYFPG5A*, where gene expression is also turned off, switching from state 1a (Figure 55a) to state 0b (Figure 55b). Other states are shown in Figure 50. The abundance of prime editing machinery is shown on the x-axis, the abundance of the *eYFPG5A* expression plasmid on the y-axis. The colourbar indicates the measured fluorescence of *eYFPG5A*. All units are in MEFL; measurements are taken 3 days after transfection. Figure 55c shows the results with complete editing machinery present. Figure 55d shows the negative control, where the editing epegRNA is missing. Figure 55e shows the fold change of *eYFPG5A* fluorescence of the edit over the negative control. As expected, the relative fluorescence decreases with increasing concentration of the editing machinery, once a certain threshold of *eYFPG5A* expression is reached. The fold change is less prominent than in Figure 52, where the baseline state has higher translation due to no uORF being present. Transfection and flow cytometry for this dataset were performed by Charles van de Mark.

In this chapter, I presented some first steps towards unlocking the potential of neuromorphic computing for synthetic biology. The first step was to build a molecular perceptron, which can accept both positive and negative inputs. I achieved this by using sequestration reactions, where ERNs serve as the negative input, and transcription is a positive input. Since multiple perceptrons are needed to form a network, I tested three orthogonal ERNs. Another key aspect of neural networks is that each connection between the molecular perceptrons must have a tuneable weight.

I explored three different strategies to tune the weights of a molecular perceptron. Firstly, by tuning the ratio of co-transfection mix between the transfection marker and either the ERN or the perceptron output. This is especially useful in a transient transfection assay, because no additional assembly is required to tune weights, enabling rapid prototyping. However, many mammalian synthetic biology applications require genome integration, making this method mainly useful for prototyping. Secondly, via inserting uORF sequences upstream of either the positive or negative input. This works as expected, since it is an established method to tune gene expression in mammalian cells (Ferreira et al., 2013). Finally, via creating a small library of *CasE* recognition site variations based on their predicted secondary RNA structure. This has the advantage that different weights have (roughly) the same sequence length, potentially facilitating downstream applications such as prime editing. While the latter two weight tuning strategies have the advantage of being compatible with single copy integration, this format of genetic weight comes with a separate issue: Since each part has distinct properties, the weight can now only be set to discrete values (unlike a continuous analogue system), which can complicate weight tuning. However, algorithms for the optimisation of discretised weights already exist in computer science (in some cases even accelerating learning), and may be used to approximate the best possible weight combination *in silico*. (In fact, the digital computers we use in our day to day lives cannot store truly analogue weights either.) (Roth et al., 2020; Gholami et al., 2021).

In addition to tuneable weights, each perceptron must also have a tuneable bias to enable neural network functionality. To demonstrate this is possible with ERN based molecular perceptrons, I tested multiple different biases via poly-transfection. Importantly, both positive as well as negative biases are possible using molecular perceptrons based on ERNs.

After I confirmed that the ERN based molecular perceptrons were fully functional, I built two simple neural networks. At first, I implemented a dual region using two molecular perceptrons. This architecture was relatively simple to build and troubleshoot for two main reasons: Firstly, this architecture only required weight tuning, since the optimal bias can be

0 for both perceptrons. Secondly, no hidden layer is necessary to compute a dual region. This made weight tuning easier, since the output of each perceptron could me measured directly in the correct context. After I manually adjusted the weights, a clear dual region was visible, using *pguCas13b* and *Csy4* based molecular perceptrons. To further demonstrate the scalability of molecular neural networks, I implemented and tuned a three perceptron network to display bandpass behaviour. This network was more difficult to tune, since it included a randomised bias, and two perceptrons were in a hidden layer. Despite the randomised bias, a clear bandpass emerged under the right genetic weights. In the future, even more complex networks may be used for difficult classification tasks that can be used in cell therapies, synthetic morphogenesis or automated differentiation pathways of stem cells (Teague et al., 2016; Guye et al., 2016; Kitada et al., 2018; Zhao et al., 2023).

While these feedforward molecular neural networks already show promise, artificial neural networks are so popular in computer science because their weights can be easily tuned, without the need to reconstruct the entire network from scratch every single time. The first step to unlock this capability in biology is to create a weight tuning strategy for biological weights, that does not rely on assembly, but instead permanently changes each genetic weight via an inducible mechanism. For this purpose, I developed a minimal working example with two genetic weights via uORFs. The mechanism I developed allows for screenable insertion as well as deletion of these uORFs via the expression of an epegRNA. Furthermore, I designed the mechanism to be repeatable, so an indefinite amount of uORFs can be inserted and deleted in a step-wise manner, enabling tuning to any genetic weight.

Moving towards an inducible system can enable feedback control during *in vivo* weight tuning, where networks that are close to the target function adjust their weights minimally, while networks that under-perform undergo significant weight modification. The weight tuning strategy I developed in this thesis can be a first step towards a biological version of backpropagation, which would unlock the potential of artificial neural networks in a fully biological context. Implementing a backpropagation algorithm in biology would require three prerequisites: Firstly, weights that are adjustable *in vivo*, based on an inducible signal. With the strategy developed here, the epegRNA to tune the biological weight could be placed under an inducible promoter. Once the promoter is activated, the epegRNA is expressed, the the weight would be adjusted into the desired direction. The second prerequisite is an appropriate cost-function, that determines how good a set of weights and biases is. Those two alone would already enable some weight optimisation strategies, where a high cost function (a bad set of weights and biases) could strongly induce epegRNA expression, whereas

a perfect set of weights and biases (where the cost-function outputs zero) would not trigger epegRNA expression at all, thereby converging to a minimised cost function, similarly to "genetic algorithms" used in computer science today. A backpropagation algorithm equivalent to the ones we are using in computer science today would also require the system to know to what extent each weight needs to be increased or decreased, as well as which weight needs adjusting (essentially doing partial differentiation *in vivo*.

4

Discussion and Conclusion

In this thesis, I explored two overarching theories: Pattern generation via the Turing mechanism, and pattern recognition via molecular neural networks. For the purpose of pattern generation, a circuit based on a previous study by Scholes et al. (2019) was implemented and tested. Initial tests in liquid culture revealed that the chosen components function as expected in the context of the circuit. Furthermore, the chosen diffusors showed some differential diffusivity, and function on a timescale that is relevant to Turing pattern formation. While initial microscopy scans in the lab resulted in a colony with stripes forming on the edge, these results were not reproducible, and the majority of my efforts went into finding a patterning condition for the circuit. While stripe patterns were not repeatable with the original circuit, a condition was found where spots of different sizes repeatedly appeared during microscopy. Additionally, a control that was not predicted to form patterns under classical Turing theory yielded a repeated stripe when imaged via confocal microscopy. For this purpose, the effect of domain growth was investigated mathematically to check for pattern formation. Despite growth typically increasing the overall parameter space for pattern formation, this was not predicted for the implemented circuit. However, under the investigation of stochastic Turing theory, parametric robustness did seem to increase with increasing growth rate. Additionally, stochastic Turing theory under the inclusion of growth also supported the deletion of a whole plasmid, which was not predicted by classical Turing theory. When this deletion was screened, repeated green stripes did indeed form.

While I can not definitively conclude that the resulting patterns were due to a classical Turing mechanism, which was the initial engineering goal, the resulting pattern is likely produced by a reaction-diffusion mechanism. This becomes evident when looking at one of the control circuits, that does not contain a diffusor linked to the circuit, which disrupts all pattern formation. Furthermore, I hypothesise that the engineered reaction-diffusion mechanism is in some form related to a Turing mechanism, possibly due to the interplay of growth and stochasticity. One subcircuit, the deletion of node C, was predicted to have increased robustness under stochastic Turing theory, as growth is increased. Screening this

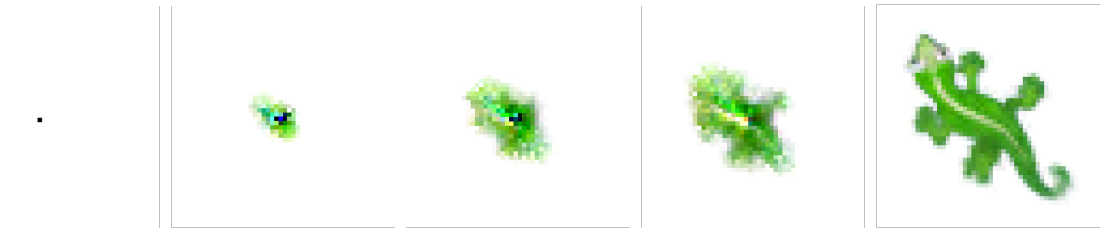
subcircuit indeed yielded a repeated stripe pattern, as predicted by theory. However, the mathematical theory I explored was not able to explain all patterns that appeared with all subcircuits. Overall, biological patterning still occurs more frequently in biology than expected - even in an engineered system.

Future work for this part of the project could focus on unifying a mathematical model and the experiments. In this thesis, the modelling was only used to test whether the engineered circuit has the potential to form Turing patterns, as well as to inform some experiments (e.g. when modelling revealed more growth can lead to outer ring addition, or that the node C deletion could form non-classical Turing patterns). A convergence of the model and the experiments could reveal what mechanism is forming the unexplained patterns. On the modelling side, this unification would require a model that keeps track of all produced RNA and protein species in the synthetic circuit. Additionally, the model might need to include a measure of cell fitness or burden. On the experimental side, each protein will have to be quantified separately, so the model can be accurately parametrised. This can be done in several ways, for example via ribosome profiling, or via quantitative mass spectrometry. Additionally, the amount of each RNA molecule per promoter in the circuit could be quantified via qPCR. While for many parameters, fitting at steady state would be sufficient, some parameters will need a dynamical fitting approach over multiple time points (e.g. determining absolute values for V_{max} and μ separately, rather than a $\frac{V_{max}}{\mu}$ ratio).

The other major aim in this thesis was to build a small molecular neural network for pattern recognition purposes. To accomplish this, three molecular perceptrons were built based on molecular sequestration reactions via the PERSIST mechanism. Sequestration reactions were picked because of their inherent ability to perform addition and subtraction inside a cell. After establishing these reactions as a foundation for molecular perceptrons that can accept both positive and negative inputs, I set out to tune the weights and biases of the molecular perceptron. For this purpose, I established three viable options to tune a molecular weight: Via modifying a co-transfection ratio, via inserting uORFs, and via modifying the ERN recognition sequence. In addition to testing the effect of each of these weight tuning mechanisms, I also established the inclusion of a molecular bias. This resulted in three tuneable molecular perceptrons, that all have the same tuning mechanisms as their counterparts in computer science. I then proceeded to build two different molecular neural networks. The first network consisted of two nodes, and produced a dual region classifier. Since no hidden layer was implemented, each node was characterised individually, and their weights were tuned to produce a diagonal region of low gene expression. This example hints

at how easy it can be to tune molecular weights to adjust classification parameters. Building on the two node network, I implemented a molecular neural network with three nodes and a hidden layer to produce a bandpass. Tuning the bandpass proved more difficult, due to two nodes being not directly observable. All weights were adjusted, and a randomised bias was included to form a diagonal bandpass. While the tuning efforts resulted in a bandpass, the signals were more noisy than in the dual region classifier. Next to a randomised bias and accumulated noise as the signal is propagated from one hidden layer to the next, this noise is due to the difficulty of manually picking optimal weights in a more complex network. This challenge to pick optimal weights showcases the need for a better weight tuning strategy. In computer science, the weights and biases of an artificial neural network are usually adjusted via learning algorithms such as genetic algorithms or backpropagation. In biology, enabling such a strategy would require an inducible way to adjust molecular weights. For this purpose, I developed a two-step strategy to insert and delete uORFs in front of a gene of interest using prime editing. The developed strategy allows for arbitrary weight adjustment of a single perceptron by expressing a specific epegRNA molecule. In the future, this strategy could be tied to a feedback mechanism, where weights are adjusted based on how well the molecular neural network performs.

Short term follow up research could focus on two main areas: Firstly, the bandpass classifier is still relatively noisy. This is in part because of the randomised bias, of which a dual region is subtracted. This noise could be reduced by finding a distinct optimal bias level, and not sampling over a random bias range. Secondly, the prime editing strategy should be confirmed via sequencing. This is especially important for the editing steps that disable gene expression, since incorrect edits that happen to turn off gene expression in an unwanted way (e.g. by introducing a frame shift mutation), cannot be differentiated from a correct edit (e.g. inserting a uORF sequence). Additionally, single cell sequencing on cells where gene expression is turned off via the first editing step could provide data on the percentage of incorrectly edited cells. Since the second step in the editing process restores gene expression, incorrect edits are unlikely to turn on gene expression, and sequencing, while still necessary for stringency, would be less informative. Further follow up research could focus on integrating the weight optimisation algorithm with a supervised learning step in order to tune the decision boundary of molecular neural network based on a target function. Another possibility is to apply the developed framework to gene therapies, such as for a cancer classifier. This would demonstrate that the inputs and outputs can be directly integrated with the natural cell states, rather than using exogenously applied DNA amounts as the inputs, and fluorescent



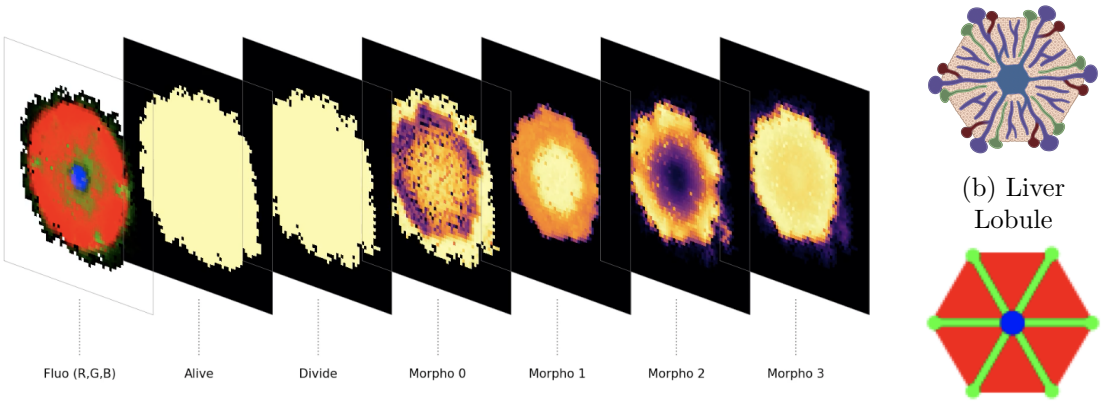
(a) t=0 (b) t=10 (c) t=20 (d) t=30 (e) t=100

Figure 56: Growing neural cellular automata model trained to grow a cartoon lizard from a single cell (Mordvintsev et al., 2020).

proteins as the outputs. Another possible application is the timing of differentiation events for the automated differentiation of stem cells. Since the change of transcription factors is gradual, analogue systems such as neural networks should outperform standard Boolean logic in this use case.

The main aim of this thesis was to develop new ways to generate and recognise gene expression patterns. My main interest here lies in the convergence of these two theories, to unlock possibilities that either pattern generation or pattern recognition cannot accomplish alone. Current research typically treats problems in morphogenesis as separate from problems in recognition and complex decision making. For example Sekine et al. (2018), who repurposed an existing Turing mechanism in a mammalian system, distinctly cite tissue engineering as a possible application for their work. Indeed, spatially organising tissue, such as being able to control differentiation of perfuse vasculature in organoids, would aid their maturation and eliminate the formation of a necrotic core. However, guiding stem cells through the correct lineages also requires precise and timed control of gene expression. This is further complicated by each individual cell having to decide when the correct timing is to proceed to the next differentiation step. Thus, only an integrated approach, requiring both pattern generation and pattern recognition, can fully unlock a new generation of synthetically grown tissue and organoids (Teague et al., 2016; Guye et al., 2016; de Souza, 2018; Zhang et al., 2021; Kim et al., 2020; Hofer and Lutolf, 2021).

Apart from integrating these two approaches in the same application, computer scientists have started to integrate reaction-diffusion principles, cell growth, and neural network structure. This lead to an alternative patterning strategy, where growing neural cellular automata form complex patterns resembling spider webs, lizards, butterflies and more (Mordvintsev et al., 2020). An example of a trained neural cellular automata model with a deep network is shown in Figure 56. In the future, these models could be adapted to use the neuromorphic components I used in this thesis, together with the morphogens I used during Turing pattern engineering. This combined approach could be used to drive useful pattern generation for



(a) Neural Cellular Automata Simulation

(b) Liver Lobule

(c) Target

Figure 57: Growing neural cellular automata model to produce a liver lobule. Four morphogens (two long range, two short range) were used to produce this model. The model was constructed by Jean Disset (2023, unpublished results). Figure 57a shows the final time point of the simulation. Figure 57b shows an abstraction of a liver lobule. Figure 57c is a simplified graphic based on a liver lobule, that was used as the simulation target.

synthetic biology. An example simulation of a biologically relevant simulation using growing neural cellular automata in conjunction with morphogens to produce a liver lobule is shown in Figure 57.

This further illustrates how the line between pattern generation and recognition is becoming more and more blurred, and how the two principles can inform and benefit each other. In the future, I hope to contribute to this blurring line.

5

Methods

In this Chapter, I give a detailed description of all the materials and methods used to produce the results presented in this thesis. The methods are described in a protocol style to facilitate reproducibility. All scripts are maintained in private GitHub repositories (profile: github.com/gkaw), and are available upon request. Plasmid sequences are available as GenBank files upon request.

5.1 Dispersion Relation

As discussed in Section 1.2.2, Turing patterns can be analysed analytically, which is computationally relatively cheap when compared to solving partial differential equations with numerical methods (Section 5.3). The analytical method used to calculate whether a stable pattern emerges from a set of equations with a given parameter combination is called dispersion relation analysis - the application of linear stability analysis to a spatial system in the frequency domain. This section explains how to derive the characteristic equation for any reaction diffusion system (Equation 3), which is used to calculate the easily interpretable plots described in Section 1.2.2 (Figure 5).

Any reaction diffusion system can be written in the general matrix notation shown in Equation 19. This method shows how the stability of any reaction diffusion system can be analysed.

$$\begin{aligned} S_i &:= \text{ [Species]} & F_i &:= \text{ Reaction term} \\ S_j &:= \text{ [Species] influencing } S_i & D_i &:= \text{ Diffusion constant of } S_i \\ S^* &:= \text{ Steady state of } S \end{aligned} \tag{19}$$

$$\frac{\partial S_i}{\partial t} = F_i\big[S_j(x,t)\big] + D_i \nabla^2 S_i(x,t)$$

The stability of a system around a certain steady state is checked by applying a small perturbation s_i to the steady state, as shown in Equation 20; a well-mixed system has reached

steady state when the rate of change is 0 ($\frac{dS_i}{dt} = F_i[S_j^*(t)] = 0$).

$$\begin{aligned} S_i(x,t) =& S_i^* + s_i(x,t) \\ s_i(x,t) =& S_i(x,t) - S_i^* \end{aligned} \tag{20}$$

Equation 21 shows the rate of change from the steady state in a well mixed system (S_i^* is eliminated when differentiating Equation 20, since it is a constant).

$$\frac{ds_i}{dt} = \frac{dS_i}{dt} \qquad \text{(since } N_i^* \text{ is constant)} \tag{21a}$$

$$= F_i\big[S_j(t)^* + s_j\big] \qquad \text{(by substitution} \tag{21b}$$

$$= F_i\big[S_j(t)^*\big] + s_i \frac{\partial F_i}{\partial S_j}\Big|_{S_j^*} + O(s_i^2) \qquad \text{(Taylor series expansion)} \tag{21c}$$

$$\approx F_i\big[S_j(t)^*\big] + s_i \frac{\partial F_i}{\partial S_j}\Big|_{S_j^*} \qquad \text{(neglecting higher order terms)} \tag{21d}$$

$$= s_i \frac{\partial F_i}{\partial S_j}\Big|_{S_j^*} \qquad \text{(since } F_i\big[S_j(x,t)^*\big] = 0) \tag{21e}$$

$$= s_i J_{ij}^* \qquad \text{(renaming the Jacobian matrix)} \tag{21f}$$

Writing the evolution of a perturbation of a steady state in the form of Equation 22 is easily interpretable, since when $\lambda > 0$ the system will evolve exponentially[1] from steady state over time; the reverse is true for negative λ. This is used to derive the characteristic equation for diffusion driven instability. To derive the characteristic equation, diffusion is reintroduced to Equation 21, resulting in Equation 23. To analyse perturbations in space, the equation is analysed in the frequency domain (enabling perturbations in the form of frequencies); the shorthand $\mathcal{F}\{f(x)\} = \tilde{f}(k)$ is used to represent the Fourier transform for readability.

$$s_i = e^{\lambda t} v \tag{22}$$

[1]The system will by definition evolve exponentially from steady state, because we are looking for a solution in the form $e^{\lambda t}v$ (Equation 22).

$$\frac{\partial s_i(x,t)}{\partial t} = s_i(x,t)J^*_{ij} + D_i\nabla^2 s_i(x,t) \quad \text{(reintroducing diffusion)} \tag{23a}$$

$$\frac{\partial \tilde{n}_i(k,t)}{\partial t} = \tilde{n}_i(k,t)J^*_{ij} - D_i k^2 \tilde{n}_i(k,t) \quad \text{(Fourier transform wrt x)} \tag{23b}$$

$$\frac{\partial}{\partial t}e^{\lambda t}v = e^{\lambda t}vJ^*_{ij} - D_i k^2 e^{\lambda t}v \quad \text{(by substitution of Eqn 22)} \tag{23c}$$

$$\lambda e^{\lambda t}v = e^{\lambda t}v(J^*_{ij} - D_i k^2) \quad \text{(differentiate and simplify)} \tag{23d}$$

$$\lambda v = v(J^*_{ij} - D_i k^2) \quad \text{(cancel out } e^{\lambda t}\text{)} \tag{23e}$$

$$v(J^*_{ij} - D_i k^2) - \lambda v = 0 \quad \text{(subtract } \lambda v \text{ from both sides)} \tag{23f}$$

$$(J^*_{ij} - D_i k^2 - \lambda I)v = 0 \quad \text{(simplify)} \tag{23g}$$

$$det(J^*_{ij} - D_i k^2 - \lambda I) = 0 \quad \text{(for nontrivial solutions, } v \neq 0\text{)} \tag{23h}$$

Zero flux boundary conditions are now applied to Equation 23 resulting in Equation 24. Note that Equation 23 results in the characteristic equation used to analyse Turing instability, as described in Section 1.2.2.

$$\mathcal{F}\{\frac{\partial}{\partial x}S_i(x,t)\} = \int_0^L \frac{\partial}{\partial x}S_i(x,t)e^{ikx}dx \tag{24a}$$

$$= [S_i(x,t)e^{ikx}]_0^L - \int_0^L S_i(x,t)(-ike^{ikx})dx \tag{24b}$$

$$= S_i(x=L,t)e^{ikL} - S_i(x=0,t) + ik\int_0^L S_i(x,t)e^{ikx}dx \tag{24c}$$

$$= ik\int_0^L S_i(x,t)e^{ikx}dx \qquad \text{(since } S_i(x=(0,L),t)=0\text{)} \tag{24d}$$

$$\therefore \mathcal{F}\{\frac{\partial^2}{\partial x^2}S_i(x,t)\} = -k^2\int_0^L S_i(x,t)e^{ikx}dx \tag{24e}$$

Rearranging Equation 24 results in Equation 25. The conclusions drawn - that the wavenumber k is restricted to discrete values based on the length of the domain L - is because at $x = 0$

and $x = L$ the rate of change is defined as 0 due to the zero flux boundary condition.

$$\mathcal{F}\{\frac{\partial^2}{\partial x^2}S_i(x,t)\} = -k^2\int_0^L S_i(x,t)cos(kx)dx - ik^2\int_0^L S_i(x,t)sin(kx)dx \quad \text{(via Euler's formula)} \tag{25a}$$

$$= -k^2\big(S_i(x,t)sin(kx) - \int_0^L \frac{\partial}{\partial x}S_i(x,t)sin(kx)dx\big) - ik^2\int_0^L S_i(x,t)sin(kx)dx \quad \text{(via integration by parts)} \tag{25b}$$

$$\therefore k = \frac{n\pi}{L}, n \in \{0, \mathbb{N}\} \quad \text{(due to BC)} \tag{25c}$$

$$\because sin(\frac{n\pi}{L}x) = 0 \text{ at } x = 0, L \tag{25d}$$

5.2 Dispersion Relation on Growing Domains

For growing domains, stability analysis was adapted from Madzvamuse et al. (2010), which is an extension of the dispersion relation shown in Section 5.1. Madzvamuse et al. (2010) show that under the inclusion of isotropic, dilational and exponential growth, the system will form a pattern with a characteristic wavelength if a newly defined parameter μ is greater than 0, and not form if $\mu < 0$. μ is defined in Equation 26, where λ is the eigenvalue of a system, and h_* is a measure of growth. For exponential, isotropic and dilational growth, h_* is defined as $h_* = mr$, where r is the growth rate, and m is the dimensionality of growth (e.g. $m = 2$ for a growing colony in 2D). Madzvamuse et al. (2010) also analyse different growth shapes (e.g. logistic growth). This however complicated analytical analysis, since h_* would not be constant throughout time, which is assumed for this analysis.

$$\mu = \lambda - h_* \tag{26a}$$

$$\therefore \lambda = \mu - h_* \tag{26b}$$

This is particularly helpful when a characteristic polynomial is available, such as the characteristic polynomial for 2 equations shown in Equation 27.

$$\det(J^*_{ij} - D_i k^2 - \lambda I) = 0 \quad \text{(Characteristic equation)} \tag{27a}$$

$$\det \begin{bmatrix} \lambda + D_A k^2 - fa & -fb \\ -ga & \lambda + D_B k^2 - gb \end{bmatrix} = 0 \quad \text{(Expand matrix)} \tag{27b}$$

$$(\lambda + D_A k^2 - fa)(\lambda + D_B k^2 - gb) - (-fb)(-ga) = 0 \quad \text{(Take determinant)} \tag{27c}$$

$$\begin{aligned} &\lambda^2 + \lambda D_B k^2 - \lambda gb + \lambda D_A k^2 \\ &+ D_A D_B k^4 - D_A k^2 gb - \lambda fa - D_B k^2 fa + fagb - fbga = 0 \end{aligned} \quad \text{(Multiply out)} \tag{27d}$$

$$\begin{aligned} &\lambda^2 + \lambda \underbrace{(D_B k^2 - gb + D_A k^2 - fa)}_{\beta} \\ &+ \underbrace{(D_A D_B k^4 - D_A k^2 gb - D_B k^2 fa + fagb - fbga)}_{\gamma} = 0 \end{aligned} \quad \text{(Simplify)} \tag{27e}$$

$$\lambda^2 + \lambda\beta + \gamma = 0 \quad \text{(Characteristic polynomial)} \tag{27f}$$

By substituting Equation 26 into Equation 27, the characteristic polynomial for any 2 equation system with exponential, isotropic and dilational growth can be derived. The resulting characteristic polynomial for 2 equations on a growing domain in shown in Equation 28.

$$(\mu + h_*)^2 + (\mu + h_*)\beta + \gamma = 0 \quad \text{(Substitute } \lambda\text{)} \tag{28a}$$

$$\mu^2 + 2\mu h_* + h_*^2 + \mu\beta + h_*\beta + \gamma = 0 \quad \text{(Multiply out)} \tag{28b}$$

$$\mu^2 + \mu(2h_* + \beta) + (\gamma + \beta h_* + h_*^2) = 0 \quad \text{(Simplify)} \tag{28c}$$

$$\mu^2 + \mu \underbrace{(2h_* + \beta)}_{B} + \underbrace{(\gamma + \beta h_*)}_{\Gamma} = 0 \quad \text{(Neglect nonlinearities)} \tag{28d}$$

$$\mu^2 + \mu B + \Gamma = 0 \quad \text{(Characteristic polynomial)} \tag{28e}$$

The resulting polynomial matches Madzvamuse et al. (2010). Additionally, if growth is set to zero ($h_* = 0$) in B and Γ, the characteristic polynomial for non-growing domains emerges. Since polynomials in this form can be solved numerically by NumPy in Python, the `numpy.roots` function was used on this polynomial for 2 equation systems (Harris et al., 2020). The same principle was then used for a 3 equation system. Because taking the

determinant of a three equation system involves a lot of terms, a new shorthand is described in Equation 29. Note that this is already a simplified version, since this analysis is specific to three nodes and two diffusors (which is the maximum analysed in this thesis).

$$\det(J^*_{ij} - D_i k^2 - \lambda I) = 0 \quad \text{(Characteristic equation)} \tag{29a}$$

$$\det \begin{bmatrix} \underbrace{\lambda + D_A k^2 - xa}_{\text{a}} & \underbrace{-xb}_{\text{b}} & \underbrace{-xc}_{\text{c}} \\ \underbrace{-ya}_{\text{d}} & \underbrace{\lambda + D_B k^2 - yb}_{\text{e}} & \underbrace{-yc}_{\text{f}} \\ \underbrace{-za}_{\text{g}} & \underbrace{-zb}_{\text{h}} & \underbrace{\lambda - zc}_{\text{i}} \end{bmatrix} = 0 \quad \text{(Expand matrix)} \tag{29b}$$

$$\det \begin{bmatrix} \text{e} & \text{f} \\ \text{h} & \text{i} \end{bmatrix} \text{a} - \det \begin{bmatrix} \text{d} & \text{f} \\ \text{g} & \text{i} \end{bmatrix} \text{b} + \det \begin{bmatrix} \text{d} & \text{e} \\ \text{g} & \text{h} \end{bmatrix} \text{c} = 0 \quad \text{(Determinant)} \tag{29c}$$

$$\text{aei} + \text{bfg} + \text{cdh} - \text{ceg} - \text{bdi} - \text{afh} = 0 \quad \text{(Determinant)} \tag{29d}$$

Because of the length of Equation 29, each of the final six terms was calculated separately, before the characteristic polynomial was calculated. Several of the terms only require minimal algebraic rearranging; these terms are shown in Equation 30.

$$\text{bfg} = (-xb)(-yc)(-za) = -xbycza \tag{30a}$$

$$\text{cdh} = (-zb)(-xc)(-ya) = -zbxcya \tag{30b}$$

$$\text{ceg} = (\lambda + D_B k^2 - yb)(-xc)(-za) = \lambda xcza + D_B k^2 xcza - ybxcza \tag{30c}$$

$$\text{bdi} = (-xb)(-ya)(\lambda - zc) = \lambda yaxb - xbyazc \tag{30d}$$

$$\text{afh} = (\lambda + D_A k^2 - xa)(-yc)(-zb) = \lambda yczb + D_A k^2 yczb - xayczb \tag{30e}$$

The first term of Equation 29 requires more restructuring to isolate λ, and is shown in Equation 31.

$$\begin{aligned}
\text{aei} =&(\lambda + D_A k^2 - xa)(\lambda + D_B k^2 - yb)(\lambda - zc) && \text{(31a)}\\
=&(\lambda - zc)(\lambda^2 + \lambda D_B k^2 - \lambda yb + \lambda D_A k^2 + D_A D_B k^4 \\
&- D_A k^2 yb - \lambda xa - D_B k^2 xa + xayb) && \text{(31b)}\\
=&\lambda^3 + \lambda^2 D_B k^2 - \lambda^2 yb + \lambda^2 D_A k^2 + \lambda D_A D_B k^4 - \lambda D_A k^2 yb - \lambda^2 xa - \lambda D_B k^2 xa \\
&+ \lambda xayb - \lambda^2 zc - \lambda zc D_B k^2 + \lambda zcyb - \lambda zc D_A k^2 - zc D_A D_B k^4 + zc D_A k^2 yb && \text{(31c)}\\
&+ \lambda xazc + zc D_A k^2 yb + zc D_B k^2 xa - zcxayb \\
=&\lambda^3 + \lambda^2 \underbrace{(D_B k^2 - yb + D_A k^2 - xa - zc)}_{\alpha} \\
&+ \lambda(D_A D_B k^4 - D_A k^2 yb - D_B k^2 xa + xayb - zc D_B k^2 + zcyb - zc D_A k^2 + xazc) \\
&+ (-zc D_A D_B k^4 + zc D_A k^2 yb + zc D_B k^2 xa - zcxayb) && \text{(31d)}
\end{aligned}$$

As a reminder, the aim was to use `numpy.roots`, which requires a polynomial in the form of Equation 32. Since Equation 31 is the only term that contains a term with λ^2, α can already be determined.

$$\lambda^3 + \alpha\lambda^2 + \beta\lambda + \gamma = 0 \tag{32}$$

To solve for β and γ, the other terms of Equation 29 must be combined. Since the equations are long, β and γ are written out separately. β is defined in Equation 33, and γ is defined in Equation 34. The underbraces indicate which term of Equation 29 the separate values come from.

$$\begin{aligned}
\beta =& \underbrace{D_A D_B k^4 - D_A k^2 yb - D_B k^2 xa + xayb - zc D_B k^2 + zcyb - zc D_A k^2 + xazc}_{\text{aei}} \\
&- \underbrace{yaxb}_{\text{bdi}} - \underbrace{xcza}_{\text{ceg}} - \underbrace{yczb}_{\text{afh}} && \text{(33a)}\\
=&k^4(D_A D_B) - k^2(D_A(yb + zc) + D_B(xa + zc)) \\
&+ (zcyb + xayb + xazc - yaxb - xcza - yczb) && \text{(33b)}
\end{aligned}$$

$$\begin{aligned}\gamma =& \underbrace{(-zcD_AD_Bk^4 + zcD_Ak^2yb + zcD_Bk^2xa - zcxayb)}_{\text{aei}} + \underbrace{(-xbycza)}_{\text{bfg}} + \underbrace{(-zbxcya)}_{\text{cdh}} \\ & - \underbrace{(D_Bk^2xcza - ybxcza)}_{\text{ceg}} - \underbrace{(-xbyazc)}_{\text{bdi}} - \underbrace{(D_Ak^2yczb - xayczb)}_{\text{afh}}\end{aligned} \tag{34a}$$

$$\begin{aligned}=& -k^4(D_AD_Bzc) + k^2(D_A(zcyb - yczb) + D_B(zcxa - xcza)) \\ & + ybxcza + zcyaxb + xayczb - zcxayb - xbycza - zbxcya\end{aligned} \tag{34b}$$

Equation 26 was then substituted into Equation 32 to arrive at a characteristic polynomial for three-equation systems with two diffusors under the consideration of exponential, isotropic and dilational growth. Higher order terms for h_* are again neglected, as in Madzvamuse et al. (2010) for two-equation systems. This results in Equation 35. Again, when h_* is set to 0, the polynomial simplifies to Equation 32, which can be used to analyse Turing instabilities without growth. The final form of Equation 35 was used with the `numpy.roots` function to determine whether Turing instabilities occur at specific growth rates (Madzvamuse et al., 2010; Harris et al., 2020).

$$\lambda^3 + \alpha\lambda^2 + \beta\lambda + \gamma = 0 \tag{35a}$$

$$(\mu + h_*)^3 + \alpha(\mu + h_*)^2 + \beta(\mu + h_*) + \gamma = 0 \tag{35b}$$

$$\mu^3 + 2\mu^2h_* + \mu h_*^2 + \mu^2h_* + 2\mu h_*^2 + h_*^3 + \alpha\mu^2 + 2\alpha\mu_* + \alpha h_*^2 + \beta\mu + \beta h_* + \gamma = 0 \tag{35c}$$

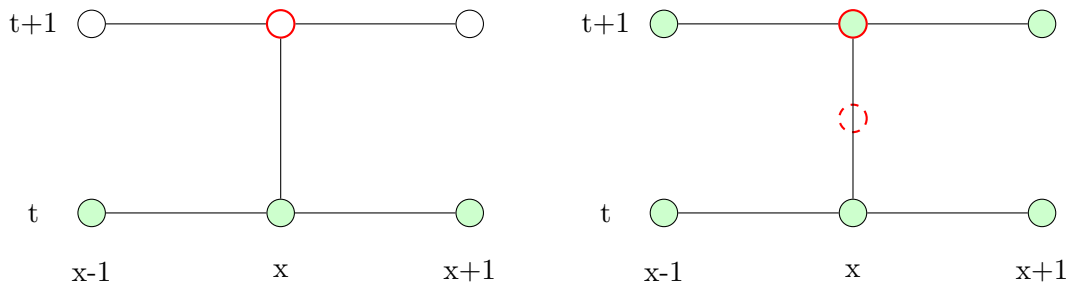
$$\mu^3 + 3\mu^2h_* + 3\mu h_*^2 + h_*^3 + \alpha\mu^2 + 2\alpha\mu_* + \alpha_*^2 + \beta\mu + \beta h_* + \gamma = 0 \tag{35d}$$

$$\mu^3 + \mu^2(3h_* + \alpha) + \mu(3h_*^2 + 2\alpha h_* + \beta) + (h_*^3 + \alpha h_*^2 + \beta h_* + \gamma) = 0 \tag{35e}$$

$$\mu^3 + \mu^2\underbrace{(3h_* + \alpha)}_{A} + \mu\underbrace{(2\alpha h_* + \beta)}_{B} + \underbrace{(\beta h_* + \gamma)}_{\Gamma} = 0 \tag{35f}$$

5.3 Numerical Solving of Partial Differential Equations

Apart from analytical methods (dispersion relation, described in Section 5.1), numerical methods can be used to solve a system of partial differential equations. Similarly to solving ordinary differential equations (ODEs) by discretising time (Equation 36), spatial systems deal with discretised space and time. The spatial system is divided into separate spatial points. ODEs are solved for each spatial point, while considering neighbouring points for diffusing molecules. These solvers are generally divided into two categories: Explicit solvers (e.g. forward difference method) and implicit solvers (e.g. backward difference method or Crank-Nicolson method). Here, two solvers (forward difference method and Crank-Nicolson



(a) Forward difference method

(b) Crank-Nicolson method

Figure 58: Spatial points at t (bottom row) and t+1 (top row). The red outlined point is the point to be calculated. Green points are used to determine the point at t+1. The red dashed outline in Figure 58b between t+1 and t is where diffusion is approximated to include all green points in the calculation.

method) are implemented and compared.

$$\frac{dS}{dt} \approx \frac{S(t+1) - S(t)}{\Delta t} \tag{36}$$

Explicit methods use values at time t to generate the spatial information for all discrete points at $t+1$. Here, the forward difference method is implemented, according to Equation 37. An illustration of this is shown in the stencil in Figure 58a. The forward difference method has the advantage that it is quick to implement, easily modified and maintained, and more easily expanded to 2D and 3D than implicit methods. However, the forward difference method has the downside that it can become unstable, and does not reach a solution. This happens when $\frac{D\Delta t}{x^2} > 0.5$, where D is the diffusion coefficient, Δt is the size of the time step, and Δx is the spacing between the discretised points (Babuska et al., 2012). This means that many more iterations are needed to reach a solution than for implicit methods, where this restriction does not apply.

$$\frac{\partial^2 S(x, t+1)}{\partial x^2} \approx \frac{S(x-1, t) - 2S(x, t) + S(x+1, t)}{\Delta x^2} \tag{37}$$

Implicit methods use the values at time $t+1$ to generate the spatial information for all discrete points at $t+1$. Here, the Crank-Nicolson method is implemented, which uses both the points at time t and the points at $t+1$ to calculate spatial effects (Crank and Nicolson, 1947). The stencil in Figure 58b shows how an average of the points at times t and $t+1$ are used to calculate the next iteration. Because the Crank-Nicolson method uses both times t and $t+1$, it is also referred to as a semi-implicit or central difference method. Equation 38 shows how the mean between these time points is used to calculate the next iteration.

Isolating all instances of species $S(t+1)$ leads to Equation 39, which can be used to calculate the next time point of S. Equation 39 can be rewritten in matrix notation, which results in a tridiagonal matrix. This can then be solved using standard numerical methods, such as `numpy.linalg.solve`. While more complicated to implement and maintain, the Crank-Nicolson method is unconditionally stable, unlike the forward difference method. This means any Δt and any Δx can be chosen, and the solver will not break due to any instabilities. However, if $\frac{D\Delta t}{\Delta x^2} >> 0.5$, the solver can display some unwanted waves, especially in earlier time steps, before steady state is reached. It is therefore recommended that $0.25 \leqslant \frac{D\Delta t}{\Delta x^2} \leqslant 1.335$ (Babuska et al., 2012).

$$\frac{\partial S}{\partial t} \approx \frac{S(x,t) + S(x,t+1)}{2}$$

$$\begin{aligned} \frac{S(x,t+1) - S(x,t)}{\Delta t} \approx \frac{1}{2}\Big[& \frac{S(x-1,t) - 2S(x,t) + S(x+1,t)}{\Delta x^2} + \\ & + \frac{S(x-1,t+1) - 2S(x,t+1) + S(x+1,t+1)}{\Delta x^2}\Big] \end{aligned} \tag{38}$$

$$\begin{aligned} -rS(x-1,t+1) + (2+2r)S(x,t+1) - rS(x+1,t+1) \approx \\ \approx rS(x-1,t) - (2+2r)S(x,t) + rS(x+1,t) \\ \text{where} \quad r := \frac{\Delta t}{\Delta x^2} \end{aligned} \tag{39}$$

$$\mathbf{A}S(t+1) = \mathbf{B}S(t) + b(t) \quad \text{where}$$

$$A = \begin{pmatrix} 2+2r & -r & & & \\ -r & 2+2r & -r & \mathbf{0} & \\ & \ddots & \ddots & \ddots & \\ & \mathbf{0} & -r & 2+2r & -r \\ & & & -r & 2+2r \end{pmatrix} S(t+1) = \begin{pmatrix} S(x=2,t+1) \\ S(x=3,t+1) \\ \vdots \\ S(x=nx-2,t+1) \\ S(x=nx-1,t+1) \end{pmatrix}$$

$$B = \begin{pmatrix} 2-2r & r & & & \\ r & 2-2r & r & \mathbf{0} & \\ & \ddots & \ddots & \ddots & \\ & \mathbf{0} & r & 2-2r & r \\ & & & r & 2-2r \end{pmatrix} S(t) = \begin{pmatrix} S(x=2,t) \\ S(x=3,t) \\ \vdots \\ S(x=nx-2,t) \\ S(x=nx-1,t) \end{pmatrix} b(t) = \begin{pmatrix} S(x=1,t) \\ 0 \\ \vdots \\ 0 \\ S(x=nx,t) \end{pmatrix} \tag{40}$$

Both the above Crank-Nicolson method, as well as the forward difference method are implemented and tested in 1D for the heat equation, as shown for species S in Equation 41 (Fick, 1855). For this, 100 spatial points were generated, and a high concentration of Species S was seeded in the middle, as shown in Equation 42. Then, diffusion is calculated for 1000 time points. Three different approaches are implemented and timed, to compare solver efficiencies. Firstly, a naive implementation of the forward difference method is tested, using a simple `for` loop structure. This has the advantage that the use of builtin methods is kept to

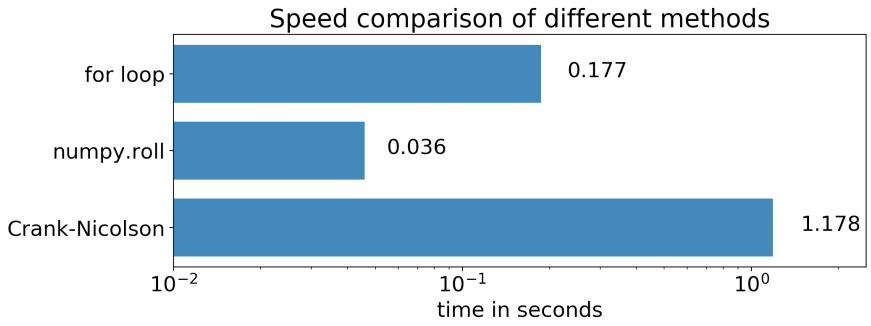


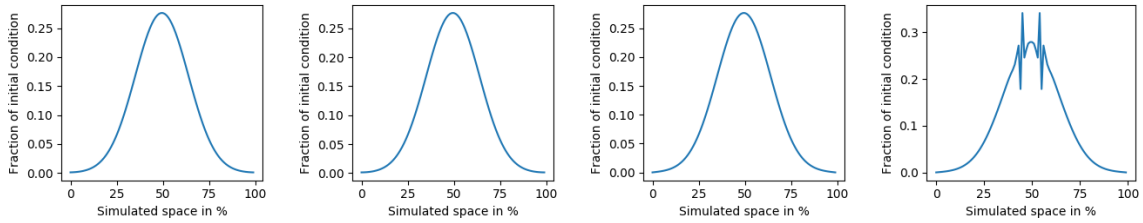
Figure 59: Speed comparison of the three tested implementations of PDE solvers on a logarithmic scale.

a minimum, which maximises control over the solver. Secondly, an optimised implementation of the forward difference method is tested, which uses the builtin `numpy.roll` method. This has the advantage that less iterations are run through a computationally expensive `for` loop, by computing all calculations on the spatial array in one step. Additionally, this method is compiled in C, which is computationally more efficient than running this complex computation in Python. Thirdly, the Crank-Nicolson method is implemented using the builtin `numpy.linalg.solve` method to solve the tridiagonal matrix shown in Equation 40. The speed comparison results for these three implementation is shown in Figure 59. The resulting times are from 100 consecutive simulations. Since the Crank-Nicolson method seems to be almost two orders of magnitude slower than the forward difference method using the `numpy.roll` method, a 100-fold higher Δt are used to simulate the same amount of time (100 times less iterations) in an attempt to outperform the explicit methods. This leads to the results shown in Figure 60. As shown, the results are identical for all implementations, apart from the Crank-Nicolson simulation that used less iterations to arrive at the same time; this result shows the unwanted oscillations that are a consequence of the large $\frac{D\Delta t}{\Delta x^2}$. Because of its speed and ease of implementation, the `numpy.roll` method is used in all simulations in Section 2. Additionally, the method can easily be scaled to 2D or 3D using Equations 43 and 44 respectively.

$$\frac{\partial S}{\partial t} = \frac{\partial^2 S}{\partial x^2} \tag{41}$$

$$\text{with } S(x=0, t=0) \text{ to } S(x=100, t=0)$$

$$S(x, t=0) = \begin{cases} 1, & 45 \leqslant x \leqslant 55 \\ 0, & x < 45, x > 55 \end{cases} \tag{42}$$



(a) For loop (b) numpy.roll (c) Crank-Nicolson (d) Fast Crank-Nicolson

Figure 60: Results comparison between different PDE solvers when simulating the heat equation in 1D. The fraction of the initial condition is plotted versus the simulated space in %. Figures 60a, 60b and 60c simulated 1000 time iterations with a Δt of 0.1. Figure 60d simulated 10 iterations with a Δt of 10.

$$\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2}S_{x,y} \approx \frac{S_{x+1,y} + S_{x-1,y} + S_{x,j+y} + S_{x,y-1} - 4S_{x,y}}{\Delta x^2} \tag{43}$$

$$\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2}\frac{\partial^2}{\partial z^2}S_{x,y,z} \approx \frac{S_{x+1,y,z} + S_{x-1,y,z} + S_{x,y+1,z} + S_{x,y-1,z} + S_{x,y,z+1} + S_{x,y,z-1} - 6S_{x,y,z}}{\Delta x^2} \tag{44}$$

Another important consideration when modelling PDEs is their boundary condition. Since the discretised lattice described above uses points at $S(x-1), S(x)$ and $S(x+1)$, the edge cases at $x = 0$ and $x = L$ (where L is the length of the simulated space) need to be defined. Generally, there are 3 different boundary conditions that are useful: Open boundary conditions, closed boundary conditions, and periodic boundary conditions. Open boundary conditions are used, when the diffusor can leave the simulated space, the simulated space is larger than the experimental setup, and the diffusor is only produced in the simulated space. In this setup, the diffusor is lost when it leaves the system. This boundary condition is implemented by padding the simulated space with 0 before applying the lattice. An example matrix in 2D is shown in Equation 45. Closed boundary conditions are used when the entire experimental setup is modelled. In this case, the chemical cannot leave the simulated space, and is kept in the system. This is achieved by keeping the derivative at $x = 0$ and $x = L$ equal to 0. Closed boundary conditions are easily implemented by padding the simulated space with the values of the outermost spatial points, thereby keeping $S(x = 0, L) - S(x = -1, L + 1) = 0$. An example matrix in 2D is shown in Equation 46. Periodic boundary conditions are used when only a small portion of the experimental setup is simulated, by approximating infinite space. In periodic boundary conditions, the chemicals leaving one side of the simulation enter the other side. This boundary condition is implemented by padding the outside of the simulated matrix with its furthest neighbour. An example matrix in 2D is shown in Equation

47.

$$\begin{matrix} & 0 & 0 & 0 & \\ 0 & S_1 & S_2 & S_3 & 0 \\ 0 & S_4 & S_5 & S_6 & 0 \\ 0 & S_7 & S_8 & S_9 & 0 \\ & 0 & 0 & 0 & \end{matrix} \quad (45) \qquad \begin{matrix} & S_1 & S_2 & S_3 & \\ S_1 & S_1 & S_2 & S_3 & S_3 \\ S_4 & S_4 & S_5 & S_6 & S_6 \\ S_7 & S_7 & S_8 & S_9 & S_9 \\ & S_7 & S_8 & S_9 & \end{matrix} \quad (46) \qquad \begin{matrix} & S_7 & S_8 & S_9 & \\ S_3 & S_1 & S_2 & S_3 & S_1 \\ S_6 & S_4 & S_5 & S_6 & S_4 \\ S_9 & S_7 & S_8 & S_9 & S_7 \\ & S_1 & S_2 & S_3 & \end{matrix} \quad (47)$$

5.4 Transfection

All transfections are performed on HEK293-T cells using a slightly modified protocol for Lipofectamine^{TM} 3000 from ThermoFisher Scientific (Invitrogen).

1. **Seed cells.** On day 0, a standard 10 cm plate with ca. 70% cell density was resuspended in fresh DMEM medium with 10% FBS and 2% non-essential amino acids (NEAA). 80,000 live cells were then seeded per well in a 24 well plate. At this stage, cells typically have a cell viability of ca. 95%. The cells were counted using an automated cell counter (Countess 3 from Thermo-Fisher Scientific). The wells were then topped off to 450-500 µl of fresh medium (DMEM, 10% FBS. 2% NEAA). The next steps were carried out the following day.

2. **Mix DNA.** All the DNA that needs to be correlated (co-transfected) was mixed at the appropriate ratios in ng.

3. **Add P3000 Enhancer.** 0.0022 µl of P3000 salts and 0.05 µl of OptiMEM were added to each co-transfection per ng of DNA. This was done by first mixing the P3000 salts and OptiMEM in bulk, vortexing well, and then adding the mixture to the DNA. It is important not to add OptiMEM and P3000 before all the DNA that should be correlated is thoroughly mixed, since this can reduce the reliability of the co-transfection ratio (Wauford et al., 2023).

4. **Add Lipofectamine 3000.** 0.0022 µl of L3000 lipids and 0.05 µl of OptiMEM were then added to the DNA mixture obtained in the previous step, and mixed well via resuspending.

5. **Incubate.** The DNA mixture was then left to incubate at room temperature for 10-15 minutes.

6. **Add transfection mix to cells.** Each finished transfection mix was then added to the well containing the cells seeded on the previous day. Seeding 80,000 cells should result

in ca. 70% confluence in a standard 24 well plate. For a poly-transfection, each well receives one DNA mix per uncorrelated input. If possible, the DNA was kept between 300 and 1000 ng per well for optimal transfection efficiency.

7. **Analyse cells.** After 2-3 days, the cells were analysed via flow cytometry (as described in Sections 5.5 and 5.6). Expression can typically already be confirmed via microscopy the day after transfection.

5.5 Flow Cytometry

Flow cytometry was carried out on a BD-FACS Fortessa II, after 2 days of transfection for all molecular perceptron experiments, and 3 days after all prime editing experiments. Laser intensities and gains were adjusted based on all single colour controls, and kept consistent between all samples in an experiment. Typically, around 350,000 events are collected for each poly-transfection, at a flow rate of ca. 1000 events per second. Gating and compensation was done in Cytoflow (Teague and Weiss, 2017–2022), as described in Section 5.6. Prior to flow cytometry, cells were prepared using the protocol below.

Materials

- Trypsin
- FACS buffer (DMEM +10% FBS, +2% NEAA, +10% BSA)
- DMEM (+10% FBS, +2% NEAA)
- PBS (phosphate buffered saline)
- 1.5 ml Eppendorf tubes
- FACS tubes (5 ml, polystyrene, with cap containing a cell strainer)
- Cells transfected 2-3 days ago in a standard 24 well tissue culture plate
- Ice bucket

Procedure

1. **Prepare cells.** First, trypsin and DMEM were prewarmed to 37 °C in a metal beads bath. FACS buffer was kept at 4 °C. Optionally, the cells were then washed with 500 µl PBS. Following the optional wash step, 500 µl of trypsin was added. The trypsin covered cells were incubated at 37 °C until they did not adhere to the bottom of the well. This typically took ca. 3 minutes. 500 µl of DMEM with FBS was then added to inhibit the trypsin reaction.

2. **Concentrate cells.** The cell suspension in each well was then transferred to a 1.5 ml Eppendorf tube. The Eppendorf tubes were then spun down for 3 minutes at 400

rcf and 4 °C in a tabletop centrifuge. The media was then aspirated, and the cells were resuspended in 250-500 µl of cold FACS buffer. Cells were passed through a cell strainer to break up any remaining clumps and doublets while transferring them to a 5 ml polystyrene FACS tube.

3. **Run flow cytometry.** Flow cytometry was then run on a BD-FACS Fortessa II at ca. 1000 events per second. Data was then analysed in Cytoflow.

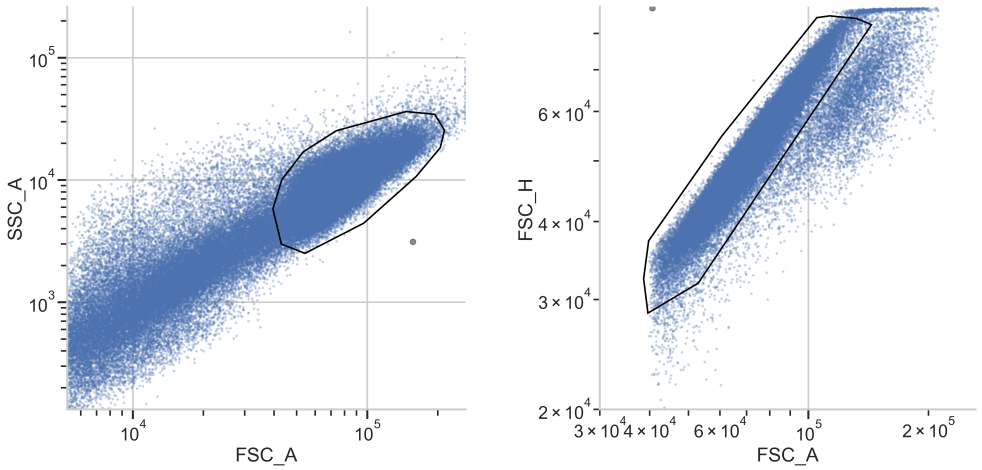
5.6 Flow Cytometry Analysis

Flow cytometry analysis was split in two parts: standard and custom analysis. Standard analysis is consistent between all flow cytometry experiments, and includes gating for healthy cell singlets, linear compensation between all colours, autofluorescence correction, beads calibration, and colour conversion into standardised units, as outlined by the TASBE workflow (Beal et al., 2019; Teague and Weiss, 2017–2022). Additionally all channels are binned using the Cytoflow package (Teague and Weiss, 2017–2022). A custom python package was developed to create all the heatmaps and statistics associated with flow cytometry shown in Section 3. This section shows a rundown of the typical flow cytometry analysis workflow. Every workflow requires at least three sets of controls: An untransfected sample, one single colour transfection for each colour used in the experiment, and a co-transfection where all colours used in the experiment are transfected at equal amounts (in ng plasmid DNA).

1. **Gating Singlets.** The first step to gating cell singlets is identifying a clear cell population in the forward scatter area (FSC-A) and side scatter area (SSC-A) channels. Healthy cells are identified in the untransfected control, to have a clear image of a healthy cell population. A gated healthy cell population is shown in Figure 61a. The healthy cell population is most easily identified on a log-log scale. The cell singlets are then separated from doublets (or more) by plotting FSC-A vs FSC-H. Cell doublets (or more) should be clearly identifiable by having a greater FSC-A/FSC-H ratio than cell singlets. An exmample gate where singlets are isolated from doublets is shown in Figure 61b.

2. **Autofluorescence Correction.** Once the cell singlets are identified, their autofluorescent properties are corrected by Cytoflow (Teague and Weiss, 2017–2022). This is again performed on the untransfected control.

3. **Linear Compensation.** Since some fluorescent protein signals bleed into other channels due to excitation and emission overlap, bleedthrough correction is performed by



(a) Cells (b) Singlets

Figure 61: Gating strategy to identify healthy cell singlets. Figure 61a shows how the untransfected control is gated to isolate the cell population in the SSC-A and FSC-A channels. Figure 61b shows how only cell singlets are included in a gate. Cell doublets appear as a separate population when plotting FSC-A versus SSC-A.

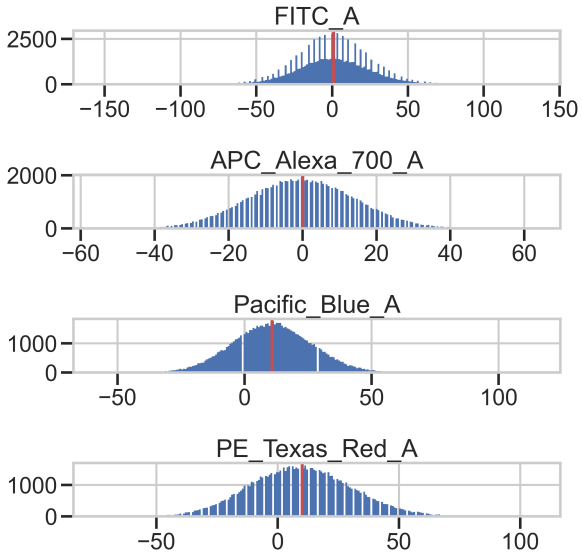


Figure 62: Autofluorescnece correction as calculated by Cytoflow for every colour used in the experiment (Teague and Weiss, 2017–2022). The autofluorescence is based on the untransfected control.

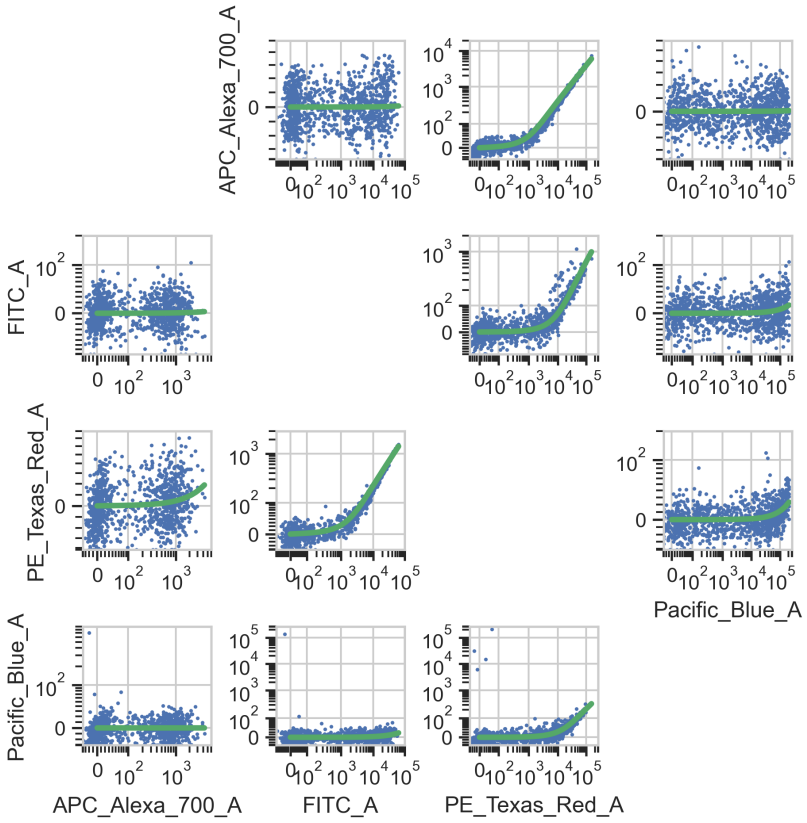


Figure 63: Linear compensation as performed by Cytoflow (Teague and Weiss, 2017–2022). The single colour controls are used to define each colour seperately, before linear compensation is performed to yield bleedthrough correction. This is done to keep all colour signals separate.

Cytoflow via linear compensation. This step requires the single colour controls, so Cytoflow can define each colour separately. Figure 63 shows the bleedthrough correction matrix for four fluorescent proteins in their respective channels: *iRFP720* in APC-Alexa-700-A, *mKate2* in PE-Texas-Red-A, *tagBFP* in Pacific-Blue-A, and *eYFPG5A* in FITC-A. As seen, bleedthrough correction is essential, especially for the pairs *iRFP720-mKate2* and *mKate2-eYFPG5A*.

4. **Beads Calibration.** After linear compensation is performed for each fluorescent protein, each channel is calibrated to standardised beads. The main advantage for this is that bead calibration should remove machine to machine variability, as well as variability between experiments; especially when the laser settings (e.g. gain) is changed between experiments. The beads used for calibration in this thesis are URCP-100-2H from Spherotech. Since no full specification sheet could be found online for these beads, the specification sheet is listed in Table 5, and was obtained by contacting Spherotech directly. The standard channels for the four channels used in this thesis

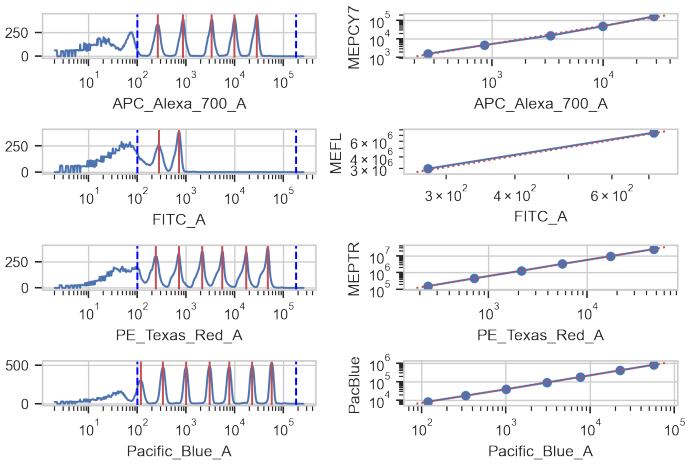


Figure 64: Example beads calibration for a four colour experiment and beads URCP-100-2H.

are: FITC-A: MEFL. Pacific-Blue-A: PacBlue. PE-Texas-Red-A: MEPTR. PE-Alexa-700-A: MEPCY7. An example beads calibration is shown in Figure 64.

5. **Colour Translation.** After the channels are calibrated to standardised beads, each channel is translated to MEFL (via the FITC-A channel). This standardises signal strength between fluorescent proteins. This is useful, so fluorescent proteins can be exchanged between experiments and co-transfections while maintaining the same results. An example that illustrates this is true is the result to determine *eYFPG5A* fluorescence, where a co-transfection of *eYFP(G5A)* with *tagBFP* yielded a slope of $1.00pm$ 2% when transfected at a 1:1 ratio (Figure 51 in Section 3.5). Figure 65 shows how the colours are linearly translated into MEFL. The two red lines on the histogram plots separates transfected from untransfected cells. The red dashed line shows the best fit linear fit to the transfected identified via a mixture model, and is used to translate each channel to MEFL (Teague and Weiss, 2017–2022).

6. **Binning.** Finally, cytoflow is used to bin all channels on a logarithmic scale. The bins are then used to calculate the statistics for the heatmaps that are displayed throughout Chapter 3. Figure 66 shows an example binning operation for the calibrated Pacific-Blue-A channel with a bin-width of 0.25 on a logarithmic scale. Each bin then maps to a value on the x-axis or y-axis for the heatmaps. For example, take a poly-transfection where *tagBFP* is correlated with *eYFPG5A* with an upstream *CasE* recognition site, and *mKate2* is correlated with *CasE*; both correlated plasmids are co-transfected with a 1:1 ratio. The cells that are in bin 7.25 for Pacific-Blue-A (in MEFL) and in bin 7.25 for PE-Texas-Red (in MEFL) should now also contain the same amount of *CasE*

Beads: URCP-100-2H								
MEFL	4648	14631	42313	128924	381106	1006897	2957538	7435549
MEPE	2800	8770	25174	74335	219816	548646	1600005	4255375
MEPTR	17518	53950	153641	450901	1283877	3254513	9431807	24840372
MEPerCP	4354	10032	22473	51739	115599	256091	562684	1201350
MEPCY5.5	2088	6705	20441	66215	211174	645020	2478405	10603147
MEeF710	2259	6862	20129	63316	196680	609247	2451473	11687960
MEPCY7	534	1555	4600	14826	47575	161926	706536	3262715
MEPTR	17518	53950	153641	450901	1283877	3254513	9431807	24840372
MEPCY5	4354	10032	22473	51739	115599	256091	562684	1201350
MEPCY5.5 2	1999	6228	20393	69124	220232	777840	2521966	8948283
MEAX700	6625	17113	58590	199825	629666	2289301	6504723	17637305
MEPCY7 2	457	1334	4666	17500	58774	230324	724800	2057002
MEAPC	1170	1970	4669	13757	36757	119744	293242	638909
MEAX680	6844	17166	56676	195246	622426	333985	6617776	17561028
MEAX700	6625	17113	58590	199825	629666	2289301	6504723	17637305
MEAPCCY7	1385	3804	13066	47512	151404	542987	1305924	2540123
PacBlue	4450	8342	17587	38906	89281	179989	408481	822214
MEAMCY	5974	10513	21623	46727	105630	213273	494395	1072308
MEPO	391	753	1797	4766	13937	39280	156244	652221
MEQ605	3133	4774	8471	16359	34465	71375	189535	517591
MEQ655	1859	2858	5598	11928	27542	66084	202508	650000
MEQ705	1695	2858	5598	11928	27542	66084	202508	650000
MEBV711	1564	3234	5516	12249	29651	71051	197915	596714
MEQ800	1358	2085	4301	10037	23446	64511	186279	644779

Table 5: Spherotech URCP-100-2H beads values for each channel used for flow cytometry beads calibration via Cytoflow (Teague and Weiss, 2017–2022). The values were obtained by contacting Spherotech directly.

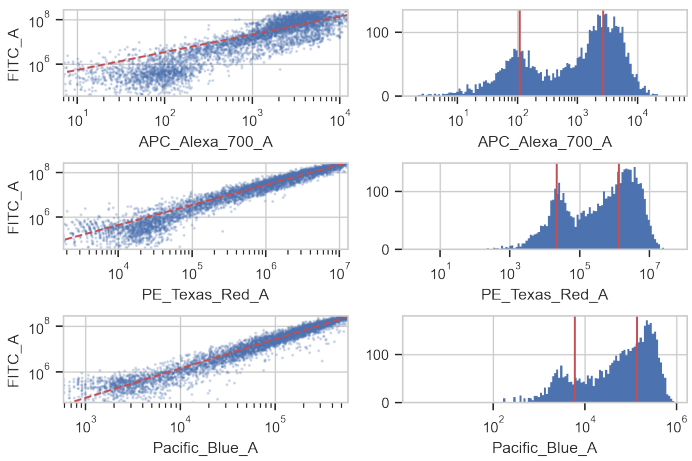


Figure 65: Colour translation from all channels into MEFL via Cytoflow (Teague and Weiss, 2017–2022).

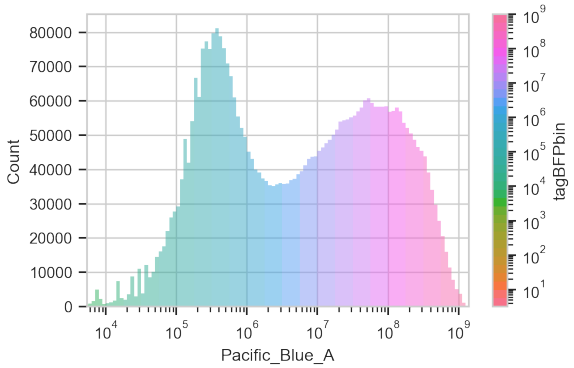


Figure 66: Binning operation on the calibrated Pacific-Blue-A channel with a bin-width of 0.25 on a logarithmic scale. Each bin is used as a value on the x-axis or y-axis for the heatmaps shown throughout Chapter 3.

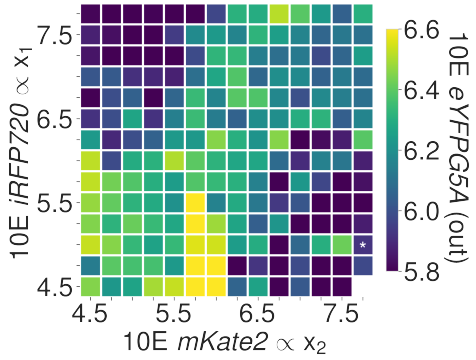


Figure 67: Example downstream custom analysis using Python and NumPy (Harris et al., 2020). Note the pixel with an asterisk in the bottom right, which was first excluded due to lacking cells, and then linearly interpolated from neighbouring data points.

and *eYFPG5A*, informing how this perceptron reacts at a 1:1 ratio by measuring the (log)mean output of *eYFPG5A* in the FITC-A channel (in MEFL). The same is true for all other bins, just that the ratio of *CasE* to *eYFP* will be different, based on their respective correlated fluorescent proteins.

7. **Downstream Analysis.** After the above standard workflow, custom python scripts were created to analyse the data. Unless otherwise stated, the log10 of the mean signal per bin was taken, as long as more than 3 cells were present in the bin. If less than three cells were present, the bin was excluded from plotting. Missing values (e.g. due to exclusion, because too few cells were present in that bin) in the selected 2D array (of two binned colours) were then linearly interpolated, to complete the image. Linearly interpolated values are marked with an asterisk in the centre of the bin throughout the thesis. An example of downstream analysis is shown in Figure 67. Note the interpolated value in the bottom right corner marked with a white asterisk.

5.7 Gibson Assembly

Gibson assembly is an enzyme based assembly method, and typically requires bespoke design and PCR reactions for every part (Gibson et al., 2009). This method was chosen for all plasmids created for Chapter 2, since only a limited number of plasmids needed to be constructed, and bespoke changes needed to be made. The following protocol outlines the materials and procedure followed to perform Gibson assembly.

Materials

- DNA templates for parts to be assembled
- Custom primers for each part amplification
- Gibson assembly MasterMix (e.g. NEB 2x HiFi Assembly Mix, NEB E5520S)
- DpnI (e.g. NEB R0176L)
- Gel extraction kit (e.g. NEB Monarch, NEB T1020L)
- High Fidelity DNA polymerase (e.g. Q5 master mix, NEB M0492L)
- Nuclease free water
- Agarose
- SYBR^{TM} Safe DNA Gel Stain (Invitrogen S33102)
- NanoDrop

Procedure

1. **Design primers.** All parts and the backbone need to be PCR amplified. When designing the primers, the position of each fragment is determined by including a 20 base pair overlap between neighbouring parts, which is used to target ligation. The overlap is included in the 5' end of the amplification primers. Apart from that, standard primer design rules should be followed.
    - Minimum primer length: 18 bp
    - A $\leqslant$ 5 °C difference should be kept between primer annealing temperatures. The annealing temperatures can be checked using an online calculator for the polymerase used (e.g. NEB tm calculator for any polymerase from NEB).
    - Recommended: The final 1-3 bases of the primer (on the 3' end) should be either guanine or cytosine.
2. **PCR amplification.** Each part in the assembly is amplified with a high fidelity polymerase (typically Q5 or Phusion from NEB) according to the respective standard protocol. After PCR amplification, DpnI is added to the reaction mix to digest the

methylated PCR template. This is important, because methylated DNA is more likely to be taken up during transformation than unmethylated (assembled) DNA. This is because plasmids are naturally methylated in *E. coli*, and unmethylated DNA is considered 'non-self'. DpnI digestion is carried out for 5-10 minutes at room temperature.

3. **Gel extract.** After running the entire reaction on an agarose gel (typically 1% agarose for 1 hour at 120 V), the band with the correct size is gel extracted with a standard gel extraction kit (e.g. NEB Monarch). The reaction is eluted in 6 µl, and the concentration is measured on a NanoDrop.

4. **Run Gibson reaction.** After determining the DNA concentration, 2.5 µl of total DNA mix is prepared in nuclease-free water including all DNA parts. All parts are included in a 2:1 excess to the backbone in a molar ratio. 2.5 µl of NEB Gibson HiFi assembly mix are then added to the DNA mixture, and incubated at 50 °C for 30-60 minutes. The Gibson reaction mix consists of four components:

   - DNA polymerase
   - 5' → 3' exonuclease
   - DNA ligase
   - dNTPs

   During the reaction, the exonuclease degrades one strand of the DNA in the 5' → 3' direction end, leaving a 3' single strand free. The 3' strand is left free to prime with its complementary strand on the insert/backbone counterpart. Once ligated, the DNA polymerase extends the 3' end until it runs out of template strand, i.e. until it reaches the region, where the template DNA becomes double stranded. The DNA ligase then joins the nicks left from the reaction, thus completing the plasmid (Gibson et al., 2009). A schematic of the reaction process is presented in Figure 68.

5. **Transformation.** After assembly, the plasmid is transformed via heatshock. Cells are then grown on the relevant antibiotic resistance, and miniprepped according to a standard DNA miniprep protocol (e.g. from Qiagen).

5.8 MoClo Assembly

MoClo assembly is a type IIS restriction enzyme based cloning method similar to Golden-Gate, that uses standardised overhangs and enzymes to denote the position and abstraction level of each part. All plasmids with a systematic prefix (e.g. pGW0042) are constructed using a slight variation from standard MoClo overhangs (Weber et al., 2011). In general,

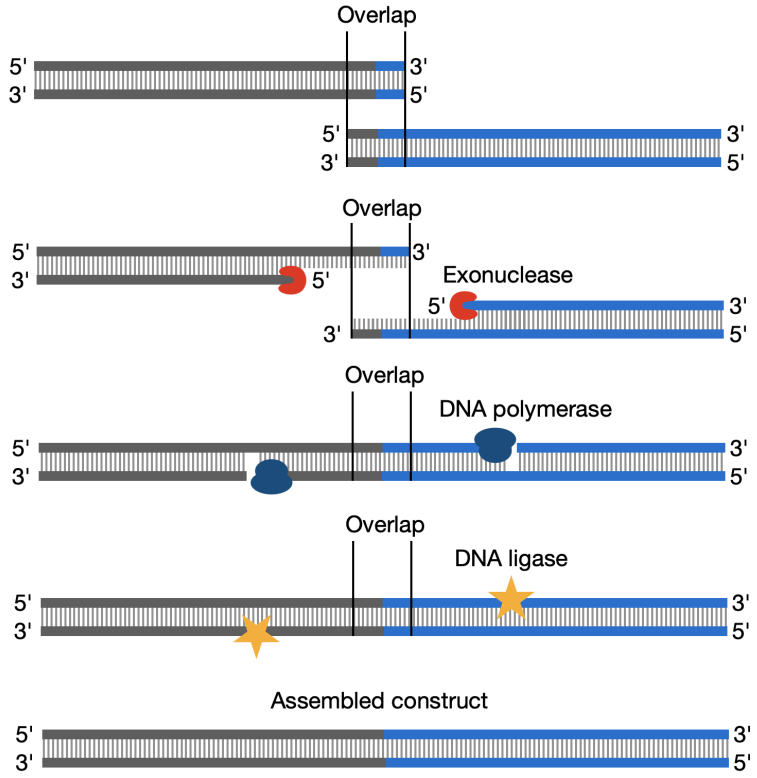


Figure 68: Schematic representation of the Gibson reaction.

there are three levels of abstraction: L0s contain the smallest standardised abstraction level, which is assigned to individual functional units. There are 6 standardised L0s, which are always ordered in the following way: Insulator, promoter, 5' regulatory sequence, gene, 3' regulatory sequence, and terminator. One of each is used and inserted into an L1 plasmid, forming the next layer of abstraction, which is a gene expression cassette. Multiple L1s can then be assembled into an L2, which is the highest level of abstraction, and is assigned to multiple gene expression plasmids. Table 6 shows the overhangs and restriction enzymes that were used to denote position and abstraction level. The following protocol outlines the materials and procedure followed to perform MoClo assembly.

Materials

- Plasmids in MoClo format
- Relevant restriction enzyme for assembly
  - BbsI to create new L0 plasmids (e.g. BbsI-HF, NEB R3539L)
  - BsaI to assemble L0 plasmids into an L1 plasmid (e.g. BsaI-HFv2, NEB R3733L)
  - SapI to assemble L1 plasmids into an L2 plasmid (e.g. NEB R0569L)
- T4 Ligase (NEB M0202L)
- T4 Ligase buffer (with ATP) (NEB B0202S)

Plasmid	Position	Function	Enzyme (donate)	5' overhang (donate)	3' overhang (donate)	Enzyme (accept)	5' overhang (accept)	3' overhang (accept)	Anitbiotic
L0 Plasmids									
L0.I	1	Insulator	BsaI	GGAG	TACT	BbsI	GGAG	TACT	Spec
L0.P.2	2	Promoter	BsaI	TACT	CAGA	BbsI	TACT	CAGA	Spec
L0.5.2	3	5' Regulation	BsaI	CAGA	AGGT	BbsI	CAGA	AGGT	Spec
L0.G	4	Gene	BsaI	AGGT	GCTT	BbsI	AGGT	GCTT	Spec
L0.3	5	3' Regulation	BsaI	GCTT	CAAC	BbsI	GCTT	CAAC	Spec
L0.T	6	Terminator	BsaI	CAAC	CGCT	BbsI	CAAC	CGCT	Spec
L1 Plasmids									
ST1-2	1	Backbone	SapI	ACC	CGC	BsaI	CGCT	GGAG	Carb
ST2-3	2	Backbone	SapI	CGC	TAC	BsaI	CGCT	GGAG	Carb
T3-XS	3	Backbone	SapI	TAC	AGA	N/A	N/A	N/A	Carb
L2 Plasmids									
pBD24.XcR	N/A	BiggyBac	N/A	N/A	N/A	SapI	AGA	ACC	Kan
pBD23.XcR	N/A	Landingpad	N/A	N/A	N/A	SapI	AGA	ACC	Kan

Table 6: Plasmid backbones used in MoClo assembly to denote the position and abstraction level of each part.

- Nuclease free water

Procedure

1. **Mix DNA.** DNA is mixed in a 1:1 molar ratio, apart from the backbone that carries the antibiotic resistance. Each part has a 2:1 excess with respect to the backbone. Every part (apart from the backbone) will have a final concentration of 3 nM in 10 µl of total reaction, where 8 µl are allocated for DNA and nuclease free water. The parts are kept in excess to the backbone to minimise the abundance of self-ligated backbone (which is the only assembly part that can survive on the antibiotics).

2. **Add enzymes.** Once the DNA mix is prepared, 2 µl of enzyme mix is added to the reaction. The enzyme mix includes 0.5 µl of the relevant restriction enzyme, which can be found in Table 6 (e.g. SapI for an L1 assembly). Additionally, 0.5 µl of T4 DNA ligase, and 1 µl of T4 DNA ligase buffer are included in the enzyme mix. The restriction enzyme recognises a sequence outside of the part to be inserted, and cuts outside of its recognition sequence, towards the part. The cut site creates a 2-3 base sticky end, which determines the position of the assembly. The overhangs are shown in Table 6. Parts that find their sticky end are ligated via the T4 ligase, which can repair DNA nicks. This reaction requires ATP, which is included in the T4 ligase buffer.

3. **Add to thermocycler.** Since T4 ligase is still active at 16 °C, whereas the restriction enzymes are only active at 37 °C, the assembly reaction can greatly benefit from cycling between those two temperatures. Cycling increases the amount of desired final product, and decreases the amount of religated plasmids. Religated plasmids still include the restriction site, and are cut every time the temperature is held at 37 °C. Since the final product does not include the restriction enzyme recognition site, the reaction is

	Step 1	Step 2	Step 3	Go to Step 2 (x99)	Step 5	Step 6	Step 7	Step 8
Degrees C	37	37	16		37	50	80	4
Time (mins)	15	5	5		15	5	5	∞

Table 7: Thermocycler settings used for MoClo assembly.

biased towards the final product. The exact thermocycler settings used are shown in Table 7. After cycling 99 times, a final digestion cuts any unwanted plasmids, before a denaturation step degrades the participating enzymes. The reaction is then kept at 4 °C for storage.

4. **Transformation.** After assembly, the plasmid is transformed via heatshock. Cells are then grown on the relevant antibiotic resistance, and miniprepped according to a standard DNA miniprep protocol (e.g. from Qiagen). Typically, 5 µl of the reaction are used in the transformation.

5.9 (e)pegRNA Assembly

(e)pegRNA plasmids were assembled via a bespoke GoldenGate method with four parts. The first part is a backbone provided by the Liu lab (Anzalone et al., 2019; Nelson et al., 2022). This plasmid contains a pU6 promoter to express RNA, and a short transcription terminator. Additionally, the plasmid contains a screenable dropout cassette (*RFP*). In case of epegRNA, the backbone also includes an RNA enhancer sequence. The other three parts are ordered as primer pairs and annealed. At least the middle primer pair has to be phosphorylated to allow annealing via T4 ligase. The following protocol was adapted from the supplementary information in Anzalone et al. (2019).

Materials

- (e)pegRNA parts as primers
  - 2 overlapping primers for the repair template
  - 2 overlapping primers for the targeting part
  - 2 primers for the (e)pegRNA scaffold (optional: order phosphorylated)
- pU6-(e)pegRNA-GG-Vector
- BsaI (e.g. BsaI-HF-v2, NEB R3733L)
- T4 Ligase (NEB M0202L)
- T4 Ligase buffer (with ATP) (NEB B0202S)
- T4 polynucleotide kinase (NEB M0201S)
- 10x rCutSmart Buffer (NEB B6004S)

- If (e)pegRNA scaffold primers were not ordered phosphorylated: T4 Polynucleotide Kinase (NEB)
- Optional: Elution Buffer (Qiagen or NEB)

Procedure

1. **Optional: Gel extract acceptor plasmid.** For optimal assembly efficiency, the pU6-(e)pegRNA-GG-Vector can be digested with BsaI, and the (ca. 2.2 kb) linearised fragment can be isolated via gel extraction using a standard gel extraction kit (e.g. NEB Monarch).

2. **Design primer pairs.** Two primer pairs need to be carefully designed for every prime editing reaction, and one standard primer pair includes the SpCas9 sgRNA scaffold sequence, which is included in every (e)pegRNA assembly.

    - The first sequence is the spacer (target) sequence, flanked by standard overhangs. The desired target's 5'→3' sequence is included on the top strand, as well as the standard 5' `CACC` and 3' `GTTTT` overhangs. The bottom strand is the reverse complement of the top strand, and includes a standard 5' `CTCTAAAAC` overhang. If the spacer sequence does not already begin with a `G`, this must be added to start transcription initiation off the pU6 promoter.
    - The second primer pair includes the (e)pegRNA extension template, and again uses standard overhangs. The RNA sense sequence is coded in the top strand, which also includes the standard `GTGC` 5' overhang. The bottom strand includes a standard 5' `AAAA` overhang for pegRNAs, and a standard `CGCG` overhang for epegRNAs. This difference is because the pegRNA ends here with a transcriptional terminator, whereas the epegRNA has another engineered RNA structure encoded in the pU6-epegRNA-GG-Vector.
    - The third pair encodes for the SpCas9 scaffold, and is standard between all (e)pegRNA assemblies. Because the scaffold is placed between primer pairs one and two, it must be 5' phosphorylated to allow for T4 ligation. The primers can either already be ordered with a 5' phosphorylation, or the 5' phosphorylation can be added enzymatically via T4 polynucleotide kinase.

    For a more detailed design description, Peyton Randolph from the Liu lab has helpfully posted a guide to design (e)pegRNAs online:
    drive.google.com/file/d/1xHWhscwIsyxUUdyGO-BZ8kleYUVmpBJA/view.

3. **Anneal primer pairs.** Each of the primer pairs is separately diluted in elution buffer.

The exact dilution is 1 µl of 100 µM top and bottom oligo, and 23 µl of elution buffer, yielding 25 µl of 4 µM double stranded DNA parts. Annealing occurs gradually in a thermocycler, where the oligos are first heated to 95 °C for 3 minutes, and are then cooled gradually at 0.1 °C per second to room temperature.

4. **Dilute Parts.** All parts are then diluted to 1 µM by adding 75 µl of nuclease free water. If the scaffold primer pair was not ordered with 5' phosphorylation, this dilution step is omitted, and instead the following phosphorylation protocol is followed.
   (a) Mix 6.25 µl of 4 µM scaffold primer pair with 0.5 µl of T4 polynucleotide kinase, 2.5 µl T4 ligase buffer, and 15.75 µl of nuclease free water.
   (b) Incubate for at least one hour at 37 °C.
5. **GoldenGate Assembly.** If the pU6-(e)pegRNA-GG-Vector is not linearised in the optional step 1, a full cycle of GolenGate assembly has to be followed as outlined in Section 5.8. If the plasmid was excised, the cycling can be reduced to 8 cycles. Figure 69 shows a visual summary of (e)pegRNA assembly. The following volumes are used for the assembly reaction:
   - 1 µl of pU6-(e)pegRNA-GG-Vector. If optional step 1 was followed, at 30 ng/µl; otherwise at 100 ng/µl.
   - 1 µl of each primer pair at 1 µM (3 pairs total).
   - 0.5 µl BsaI
   - 0.5 µl T4 DNA ligase
   - 1 µl 10x T4 DNA ligase buffer
   - 4 µl nuclease free water (adding up to a total reaction volume of 10 µl).
6. **Transformation.** After assembly, the plasmid is transformed via heatshock. Cells are then grown on the relevant antibiotic resistance, and miniprepped according to a standard DNA miniprep protocol (e.g. from Qiagen). Typically, 5 µl of the reaction are used in the transformation.

5.10 Electrotransformation

Before all microscopy experiments outlined in Chapter 2, cells were transformed using electroporation according to the following protocol.

Materials

- SOC outgrowth medium (e.g. Formedium SOC0202)
- MK01 electrocompetent cells (Kogenaru and Tans, 2014)

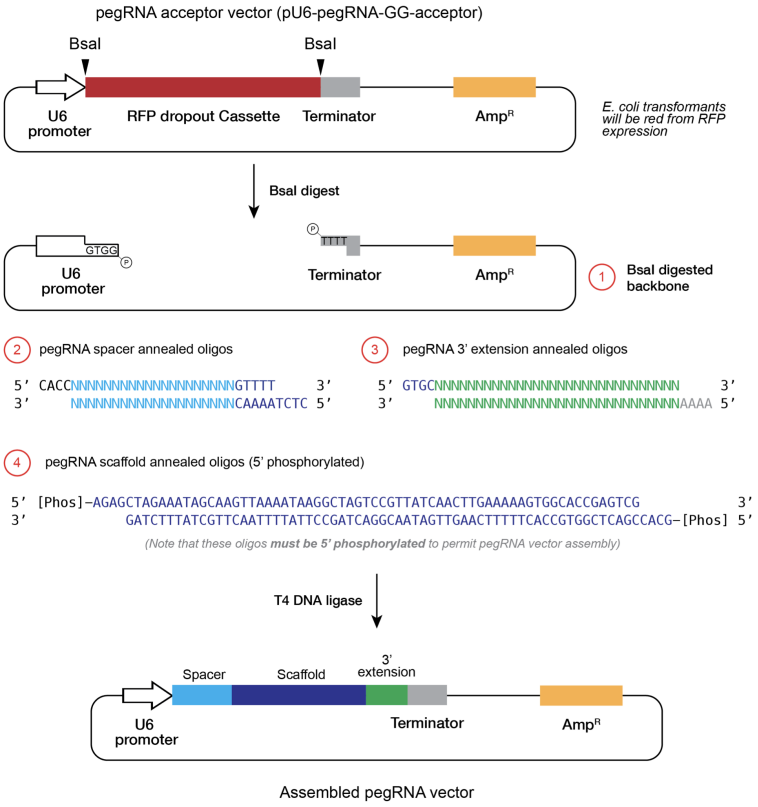


Figure 69: pegRNA cloning by GoldenGate assembly. The figure is taken from the supplementary information from Anzalone et al. (2019), and shows the example for the case of pegRNA assembly, not epegRNA assembly (which only differes in a short sequence in one of the primer pairs, as indicated in the main text).

- Plasmids (10 µg / ml stock)
- LB-Agar plate with appropriate antibiotics
- 14 ml aeration tube
- 1.5 ml Eppendorf tube
- 1 mm electroporation cuvette

Procedure

1. **Set up.** Before electroporation, plates and SOC were prewarmed to 37 °C, and the electroporation cuvette was cooled on ice. 1 µl of DNA (if two plasmids: 2x0.5 µl) were mixed in a 1.5 ml Eppendorf tube, and stored on ice. 25 µl of MK01 competent cells were also gently thawed on ice.
2. **Electroporation.** Once the cells were thawed, 25 µl of cells were gently resuspended on top of the DNA and transferred to the electroporation cuvette on ice. A pipette was preloaded with 950 µl of SOC outgrowth medium. The cells were then electroporated at 100 Ω, 25 µF, and 1.8 kV. If the electroporator showed a reading between 4.8 and 5.0, the electroporation was likely successful. Between 4.6 and 4.8, an additional plate with higher volume of cells after outgrowth was plated. If the reading was $<$ than 4.6 the transformation was repeated. Extra care was taken that the contact plates on the cuvette were dry during electroporation.
3. **Outgrowth.** Immediately after electroporation, the cells were resuspended in 950 µl SOC outgrowth medium, and incubated for 1 hour at 37 °C and 220 rpm (single orbital) in a 14 ml aeration tube. 20 µl of cell suspension was then spread on an LB-Agar plate with appropriate antibiotics and growth over night.

5.11 Heat Shock Transformation

Heat shock transformation was used for every transformation where only a single plasmid needed to be taken up by the cells (e.g. for downstream miniprep). This protocol was used after every assembly step, and was never used to prepare cells for microscopy.

Materials

- Chemically competent cells (e.g. commercial *Stellar* cells)
- SOC outgrowth medium
- Established plasmid or assembly mix
- PCR tubes (for competent cells)
- 14 ml aeration tubes

Procedure

1. **Set up.** Before heat shock, plates and SOC were prewarmed to 37 °C, and 50 µof competent cells were gently thawed on ice. DNA was gently mixed with the competent cells. Depending on the upstream reaction, different amounts of DNA were used: 5 µl of MoClo reaction, 2 µl of Gibson reaction, or 1-25 ng of established plasmid DNA. The cells and DNA were mixed either in a PCR tube, if the heat shock was carried out in a thermocycler, or in a 1.5 ml Eppendorf tube, in case a heat block or water bath was used.

2. **Heat shock.** The cells with DNA were incubated on ice for 10 minutes (or at 4 °C in a thermocycler), heated to 42 °C for 45 seconds, and then put back on ice (or 4 °C in a thermocycler) for 5 minutes.

3. **Outgrowth.** Following heat shock, the cells were resuspended in 950 µl SOC outgrowth medium, and incubated for 1 hour at 37 °C and 220 rpm (single orbital) in a 14 ml aeration tube. 20 µl of cell suspension was then spread on an LB-Agar plate with appropriate antibiotics and growth over night.

5.12 Cell Strains Used

The three *E. coli* strains shown in Table 8 were used to make competent cells.

Strain	**Source**	**Genotype**
Stellar	Takara Bio (#636763)	Parental strain: HST08. Genotype: F−, endA1, supE44, thi−1, recA1, relA1, gyrA96, phoA, ϕ80d lacZΔ M15, Δ(lacZYA - argF) U169, Δ(mrr - hsdRMS - mcrBC), ΔmcrA, λ−
TOP10	Invitrogen (C404010)	F−mcrA Δ(mrr-hsdRMS-mcrBC) ϕ80lacZΔM15 ΔlacX74 recA1 araD139 Δ(araleu)7697 galU galK rpsL (StrR) endA1 nupG
MK01	Kogenaru and Tans (2014)	Parental strain: BW27783. Genotype: F−Δ(araD-araB)567, ΔlacZ4787(::rrnB-3), λ−, Δ(araH-araF)570(::FRT), ΔaraEp-532::FRT, φPcp8araE535, rph-1, Δ(rhaD-rhaB)568, hsdR514

Table 8: Summary of the *E. coli* strains used for transformation.

5.13 Make Electrocompetent Cells

The whole protocol was carried out on ice. Since no antibiotic resistance gene is present in the cells at this stage, extra attention was given to aseptic technique, and the entire protocol was performed in a laminar flow hood. To minimally stress the cells and retain

competence, the entire workflow was performed under 60 minutes. All electrocompetent cells were of the MK01 strain to maintain consistency between microscopy experiments. For further information on the cell genotype, see Table 8. To increase transformation efficiency, the pCC1 marionette receptor cassette can be pretransformed in the original MK01 seed cells (Kogenaru and Tans, 2014). In this case, all media was supplemented with 25 µg/ml of Chloramphenicol. The volumes listed below should produce ca. twenty 25 µl reactions.

Materials

- 250 ml 2xTY medium (Sigma-Aldrich Y1003)
- 100 ml 10% glycerol (e.g. Sigma-Aldrich G5516)
- 300 ml MilliQ water
- MK01 seed cells (Kogenaru and Tans, 2014)
- SOC outgrowth medium (e.g. Formedium SOC0202)
- LB agar plate (no antibiotics or Chloramphenicol)
- 1.5 ml Eppendorf tubes
- 14 ml aeration tubes
- 50 ml sterile falcon tubes stored at -20 °C
- 500 ml Erlenmeyer flask
- Dry ice
- Ethanol

Procedure

1. **Isolate single MK01 colonies.** If the the marionette receptor cassette was included, previously competent MK01 cells were transformed via electroporation according to the protocol in Section 5.10. Otherwise single colonies were obtained by streaking on an LB-agar plate without added antibiotic. The plate was then incubated at 37 °C over night. A single colony was then picked, and grown overnight at 37 °C in 2 ml of 2xYTσ medium in a 14 ml aeration tube. Additionally, 250 ml of 2xYTσ medium was stored at 37 °C overnight in an Erlenmeyer flask.
2. **Grow cells to OD600 0.4 - 0.6.** 1 ml of the overnight pre-culture was transferred to the warm 250 ml 2xYT in an Erlenmeyer flask. The cells were then incubated at 37 °C at 220 rpm. The OD600 was checked intermittently until the OD600 was between 0.4 and 0.6. This usually took around 2 hours. During the incubation time, the centrifuge was cooled to 4 °C, and an ice bucket aswell as a dry ice bucket were prepared by sterilising the surface with ethanol.

3. **Concentrate cells.** Once the correct OD600 was reached, the Erlenmeyer flask was transferred to an ice bucket, and the cells were cooled for 5 minutes. Four 50 ml cold falcon tubes were filled with the cooled cell culture, and spun down for 5 minutes at 4 °C and 3000 rcf. The four cell pellets were then resuspended in a total of 100 ml MilliQ water in two chilled 50 ml falcon tubes, halving the total volume and washing the cells. The cells were then again spun down for 5 minutes at 4 °C and 3000 rcf. After the wash step, the cells were resuspended in a total of 100 ml of 10% glycerol. The cells were then spun down for 7 minutes at 4 °C and 3000 rcf, and the supernatant was discarded. During centrifugation, the Eppendorf tubes were chilled on dry ice.
4. **Flash freeze aliquots.** The supernatant was discarded, and cells were resuspended in the remaining 10% glycerol. The cells were then flash frozen in 50 µl aliquots, to minimize re-freezing. The aliquots were then stored at -80 °C.

5.14 Make Heat Shock Competent Cells

The whole protocol was carried out on ice. Since no antibiotic resistance gene is present in the cells at this stage, extra attention was given to aseptic technique. All chemically competent cells were of the TOP10 strain (Invitrogen, C404010) in the Isalan lab at Imperial College London, and of the Stellar cell strain in the Weiss lab at MIT. Stellar cells were bought and used directly. The following protocol is exclusively for TOP10 cells. For further information on the cell genotypes, see Table 8.

Materials

- TfbI (Ingredients from Sigma-Aldrich)
  - 30 mM potassium acetate
  - 100 mM rubidium chloride
  - 10 mM calcium chloride dihydrate
  - 50 mM manganese chloride
  - 15% glycerol
  - Acetic acid (to adjust pH to 5.8)
  - Solvent: MilliQ
- TfbII (Ingredients from Sigma-Aldrich)
  - 10 mM MOPS (3-(N-morpholino)propanesulfonic acid)
  - 75 mM calcium chloride
  - 10 mM rubidium chloride
  - 15% glycerol

  - Sodium hydroxide (to adjust pH to 6.5)
  - Solvent: MilliQ
- 100 ml LB medium
- 100 ml 10% glycerol (e.g. Sigma-Aldrich G5516)
- 300 ml MilliQ water
- TOP10 seed cells (Invitrogen C404010)
- SOC outgrowth medium (e.g. Formedium SOC0202)
- LB agar plate
- 1.5 ml Eppendorf tubes
- 14 ml aeration tubes
- 50 ml sterile falcon tubes stored at -20 °C
- 500 ml Erlenmeyer flask
- Dry ice
- Ethanol

Procedure

1. **Isolate single TOP10 colonies.** Single colonies were obtained by streaking on an LB-agar plate without added antibiotic. The plate was then incubated at 37 °C over night. A single colony was then picked, and grown overnight at 37 °C in 2 ml of LB medium in a 14 ml aeration tube. Additionally, 100 ml of LB medium was stored at 37 °C overnight in an Erlenmeyer flask.
2. **Grow cells to OD600 0.4 - 0.6.** 1 ml of the overnight pre-culture was transferred to the warm 100 ml LB medium in an Erlenmeyer flask. The cells were then incubated at 37 °C at 220 rpm. The OD600 was checked intermittently until the OD600 was between 0.4 and 0.6. This usually took around 2 hours. During the incubation time, the centrifuge was cooled to 4 °C, and an ice bucket as well as a dry ice bucket were prepared by sterilising the surface with ethanol.
3. **Wash in transformation buffers.** Once the correct OD600 was reached, the Erlenmeyer flask was transferred to an ice bucket, and the cells were cooled for 15 minutes. Four 50 ml cold falcon tubes were filled with the cooled cell culture, and spun down for 5 minutes at 4 °C and 3000 rcf. The four cell pellets were then resuspended in a total of 40 ml TfbI buffer, and incubated on ice for 15 minutes. The cells were then again spun down for 5 minutes at 4 °C and 3000 rcf. After the TfbI wash step, the cells were resuspended in a total of 4 ml of TfbII buffer and incubated on ice for 15 minutes.

4. **Flash freeze aliquots.** After incubation in TfbII, the cells were flash frozen in 50 µl aliquots. The aliquots were then stored at -80 °C.

5.15 Microscopy

Prior to microscopy, MK01 *E. coli* cells were transformed via electroporation (see Section 5.10) with the relevant plasmids. Confocal microscopy was carried out in the Facility for Imaging my Light Microscopy (FILM) at Imperial College London on a Leica SP8. FILM is part-supported by funding from the Wellcome Trust (grant 104931/Z/14/Z) and BBSRC (grant BB/L015129/1). Wide field microscopy was carried out on a Zeiss Cell Observer microscope in the Isalan lab at Imperial College London.

Materials

- Standard 12 well tissue culture plate (e.g. Falcon 12-well, REF 353043)
- Transformed *E. coli*
- Parafilm
- Relevant Medium and inducers

1. **Prepare plate.** The 12 well plates were prepared with different media types, agar thicknesses and inducer concentrations. For media types, inducer concentrations and agar thicknesses, see Section 2.4. Once the plate was prepared with the correct inducers, a colony was picked from an overnight plate with a toothpick. The MK01 *E. coli* cells were transformed the day before. If the cells were grown on very high ATC levels (>10 µM), the cells were preconditioned by transforming onto an agar plate containing 10 µM of ATC to better deal with the toxicity of ATC at high concentrations.
2. **Incubate cells.** Plates were then wrapped in parafilm to prevent drying out of the agar, and covered in a layer of aluminium foil to prevent photobleaching of the fluorescent proteins (in case the signal of the emergent pattern is very weak). The plates were then incubated at 37 °C for ca. 4 days.
3. **Image colonies.** The colonies were then imaged in either the Leica SP8 microscope, or the Zeiss Cell Observer microscope. The large microscopy scans were all carried out on a wide field microscope for higher throughput. The imaging area of both microscopes was heated to 37 °C during imaging. Parameters for imaging on the Leica SP8 were kept identical to the settings used by Jure Tica and Tong Zhu, which were the settings used to capture the repeated stripe pattern shown in Section 2.4. Parameters provided by Jure Tica. The wide field microscopy settings were optimised for faster screening

Type	Parameter	Value
Objective	Name	HC PL APO CS2 (dry)
	Magnification	10x
	Numerical aperture	0.4
Fluorescence GFP	Excitation wavelength	488 nm
	Laser, power	Argon, 15%
	Laser fractional intensity	1%-4%
	Detector	HyD
	Emission wavelength	495 - 560 nm
	Detector gain	10% - 200%
Fluorescence mCherry	Excitation wavelength	561 nm
	Laser	DPSS 561
	Laser fractional intensity	1%-4%
	Detector	HyD
	Emission wavelength	580 - 780 nm
	Detector gain	10% - 200%
Image acquisition	Sequential scan	Between lines
	Speed	400 Hz
	Format	1024 x 1024 (mostly)
	Bit depth	16
	Line average	4
	Frame average	1
	Zoom factor	Adjusted as needed
	Optical section thickness	7.2 µm
	Pinhole	1 AU (Airy 1)

Table 9: Microscope parameters for fluorescence imaging using the Leica SP8 provided by FILM.

and setup. Parameters for the Leica SP8 are shown in Table 9. Parameters for the Zeiss Cell Observer are shown in Table 10.

5.16 Liquid Culture Experiments

All liquid culture experiments in Chapter 2 for circuit characterisation were carried out on a TECAN Spark microplate reader. The parameter settings for the plate reader can be found in Table 11.

Materials

- Flat bottom microplate (e.g. from Greiner Bio-One, 655096)
- 2xYT medium (Sigma-Aldrich Y1003)
- 50 ml Falcon tube
- Relevant inducers

1. **Grow cell culture.** *E. coli* (strain MK01) were transformed via electroporation (Section 5.10) the day before the experiment should be carried out. A single colony was

Type	Parameter	Value
Objective 1	Name	EC PlnN Ph1 DICO (dry)
	Magnification	5x
	Numerical aperture	0.16
Objective 2	Name	EC PlnN Ph1 DICI (dry)
	Magnification	10x
	Numerical aperture	0.3
Fluorescence GFP	Excitation	BP 545/25
	Laser	HXP 120V
	Laser fraction	40%
	Detector	Hamamatsu Camera
	Emission wavelength	BP 605/70
	Filter set	Zeiss 43 (000000-1114-101)
Fluorescence mCherry	Excitation	BP 470/40
	Laser	HXP 120V
	Laser fraction	40%
	Detector	Hamamatsu Camera
	Emission wavelength	BP 525/50
	Filter set	Zeiss 38 (000000-1031-346)
Image acquisition	Binning	4x4
	Format	2048x2048 (mostly)
	Focus strategy	Multi-region Z values
	Zoom factor	Adjusted as needed
	Exposure Time	Automatic

Table 10: Microscope parameters for fluorescence imaging using the Zeiss Cell Observer wide field microscope.

Type	Parameter	Value
Kinetic cycle	Duration	18 hours
	Interval time	10 minutes
Absorbance	Wavelength	600 nm
	Number of flashes	10
	Settle time	50 ms
Fluorescence GFP	Type	Top reading, filter
	Excitation wavelength	485 nm
	Excitation bandwidth	20 nm
	Emission wavelength	535 nm
	Emission bandwidth	25 nm
	Gain	25
	Mirror	Dichroic 510
Fluorescence mCherry	Type	Top reading, filter
	Excitation wavelength	580 nm
	Excitation bandwidth	20 nm
	Emission wavelength	635 nm
	Emission bandwidth	35 nm
	Gain	30
	Mirror	50% mirror
Fluorescence, other	Number of reads	30
	Integration time	40 µs
	Lag time	0
	Settle time	0
	Z-position	17806 µm
Shaking	Type	Double orbital
	Duration	Continuous
	Amplitude	2 mm
	Frequency	150 rpm

Table 11: Parameters used in the TECAN Spark to measure fluorescence of the circuit. The parameter values were copied from a method set up by Jure Tica.

then picked, and incubated at 37 °C and 220 rpm (single orbital) in ca. 15 ml of 2xYT medium in a 50 ml Falcon tube until an OD600 of 0.5-0.8 was reached.

2. **Prepare microplate.** Each well in the outermost row/column of the plate was filled with 150 µl of 2xYT medium. All other wells received 10 µl of each inducer at the relevant concentration diluted in 2xYT medium (typically two inducers were added per well). Each well containing the inducer mix was then topped up to 150 µl with the cell suspension.

3. **Transfer to TECAN Spark.** The 96 well plate was then transferred to the TECAN Spark, and left to measure overnight. The temperature was set to 37 °C.

5.17 Diffusion Characterisation

Diffusion characterisation experiments described in Section 2.3 were carried out on a TECAN Spark, with the parameters shown in Table 12.

Materials

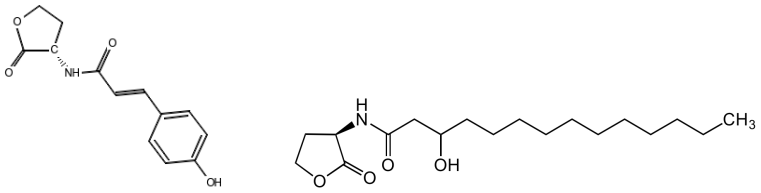
- 6 well plate (e.g. Greiner 657160)
- Agar
- 2xYT medium (Sigma-Aldrich Y1003)
- Relevant inducers

Procedure

1. **Grow cell culture.** *E. coli* (strain MK01) were transformed via electroporation (Section 5.10) the day before the experiment. A single colony was then picked, and incubated at 37 °C and 220 rpm in ca. 15 ml of 2xYT medium in a 50 ml Falcon tube until an OD600 of 0.5-0.8 was reached.
2. **Prepare 6 well plate.** A base of 1 ml of hard 1.4% agar was poured in a standard size 6 well plate. Inducer was then added to the hard agar. When no inducer was added to a well, the agar was supplemented with 2xYT medium to keep the agar concentration consistent between wells. The *E. coli* cell culture was then mixed 1:1 with the hard agar including the inducers, yielding in a 0.7% cell suspension. 1 ml of the 0.7% agar suspension of *E. coli* cells was then distributed on top of the hard agar.
3. **Transfer to TECAN Spark.** The 6 well plate was then transferred to the TECAN Spark, and left to measure overnight. The temperature was set to 37 °C.

Type	**Parameter**	**Value**
Kinetic cycle	Duration	20 hours
	Interval time	20 minutes
Fluorescence GFP	Gain	40
Fluorescence mCherry	Gain	40
Fluorescence, other	Number of flashes	2
	Z-position	29200 µm
Multiple reads per well	Shape	Circle, filled
	Grid size	15x15
	Border	1500 µm
Shaking	Type	No shaking

Table 12: Parameters used in the TECAN Spark for the diffusion characterisation experiments shown in Section 2.3. The parameter values were copied from a method set up by Jure Tica.



(a) pC-HSL (b) 3OHC$_{14}$HSL

Figure 70: HSL molecules used in this thesis.

5.18 Figure Design

All figures with abstracted protein or nucleotide graphics were created with BioRender.com under an academic Lab licence purchased by the Weiss lab. A publication license has been granted to Imperial College London (Spiral) to publish and distribute these figures for non-commercial use. The license agreement confirmation is available upon request. The following figures used BioRender graphics: 6, 7, 9, 11, 37, 45b, 47b, 49, 57, 68.

Plasmid representations in SBOL format were plotted using DNAplotlib (Der et al., 2017).

5.19 Parts Used

Table 13 shows a description of all functional parts used throughout this thesis. The definitive sequence of each part can be found in the source listed in the third column. These parts were assembled into the plasmids shown in Section 5.21, which is a complete list of all plasmids used throughout the thesis. Additionally, Figure 70 shows the molecular structure of both signalling molecules used throughout Chapter 2.

Table 13: Part abbreviations used throughout the thesis, as well as their respective description and sequence source.

Abbreviation	**Full name**	**Description**	**Source**
Ptac		Constitutive promoter	Meyer et al. (2019)
*Prpa**	pC-HSL activated promoter	Promoter activated by *rpaR* when pC-HSL is present.	Du et al. (2020)
Pcin	3OHC$_{14}$-HSL activated promoter	Promoter activated by $cinR^{AM}$ when 3OHC$_{14}$-HSL is present.	Meyer et al. (2019)
lacO	lac operator	Binding site for *lacI*, which will repress the upstream *Ptac* promoter. Sequence: `AATTGTGAGCGGATAACAATT`.	Meyer et al. (2019)

Table 13 continued from previous page

Abbreviation	Full name	Description	Source
rpaI	First synthesis enzyme of pC-HSL.	Synthesis enzyme for pC-HSL. pC-HSL binds the *rpaR* transcription factor to activate gene expression of *Prpa*. This enzyme is the first of three enzymes in the pC-HSL synthesis chain. The other two enzymes *4cI* and *tal* are always expressed constitutively on the same plasmid (pCOLA-nodeA). Endogeneous to *Rhodobacter palustris*.	Du et al. (2020)
4cI	Second synthesis enzyme of pC-HSL.	Endogeneous to *Rhodobacter palustris*.	Du et al. (2020)
tal	Third synthesis enzyme of pC-HSL.	Endogeneous to *Rhodobacter sphaeroides*.	Du et al. (2020)
tetR	tet repressor	Transcription factor (repressor). Binds a *tetO* site downstream of a constitutive promoter to repress its transcription. This interaction can be stopped by adding ATC.	Meyer et al. (2019)
tetO	tet Operator	Operator. Bound by *tetR* to repress an upstream promoter. This interaction can be stopped by adding ATC. Sequence: `TCCCTATCAGTGATAGAGA`.	Meyer et al. (2019)
cinI	$3OHC_{14}$-HSL synthase	Synthesis enzyme for $3OHC_{14}$-HSL. $3OHC_{14}$-HSL binds the $cinR^{AM}$ transcription factor to activate gene expression of *Pcin*.	Meyer et al. (2019)
lacI	lac repressor	Transcription factor (repressor). Binds a *lacO* site downstream of a constitutive *Ptac* promoter to repress its transcription. This interaction can be stopped by adding IPTG. Future circuit implementations could use the optimised $lacI^{AM}$ transcription factor.	Meyer et al. (2019)
cI	lambda phage cI repressor	*cI* is a dual transcription activator / repressor, depending on the directionality of the double operator site. In this thesis, the wildtype cI is only used as a transcriptional repressor.	Brödel et al. (2016)
cIO	lambda phage cI operator (PMO3CS)	Binding site for the wild type cI repressor. Sequence: `TATCACCGCCGGTGATA`.	Brödel et al. (2016)

Table 13 continued from previous page

Abbreviation	Full name	Description	Source
cI*	*cI5G6G,P* lambda phage dual transcription factor variant	*cI5G6G,P* is evolved via phagemid directed evolution, and acts as a dual activator / repressor. Two operator sites are required for activation, which are spaced 7nt apart.	Brödel et al. (2016)
cIO*	*cI5G6G,P* binding site (PMO15G6G and PMO25G6G used)	One binding sites is used in a single promoter for activation (PMO15G6G followed by PMO25G6G on node C), one for repression (only PMO15G6G on node B). Sequence (PMO15G6G): `TATCGGCGCCGCCGATA`. Sequence (PMO25G6G): `CAACGGCGGCGCCGATA`.	Brödel et al. (2016)
PcI*	Modified P_{RM} promoter	Promoter activated by cI*. Contains operators PMO15G6G followed by PMO25G6G. Sequence (PMO15G6G): `TATCGGCGCCGCCGATA`. Sequence (PMO25G6G): `CAACGGCGGCGCCGATA`.	Brödel et al. (2016); Chen et al. (2018); Cox et al. (2007)
mCh	*mCherry*	Red fluorescent protein. Excitation: 587 nm. Emission: 610 nm.	GenBank: AAV52164
GFP	*sfGFP*	Superfolded green fluorescent protein. Excitation: 485 nm. Emission: 510 nm.	Storch et al. (2015)
rpaR		Receptor for pC-HSL. Binds the *Prpa* promoter, activating gene expression when pC-HSL is present. Endogeneous to *Rhodobacter palustris*.	Du et al. (2020)
$cinR^{AM}$	Receptor for $3OHC_{14}$-HSL.	Receptor was improved from the wild type for a better dose response.	Meyer et al. (2019)
AiiA	N-acyl homoserine lactonase	Degrades both HSL molecules (pC and $3OHC_{14}$) used in the circuit 3954 intracellularly.	Meyer et al. (2019)
Eukaryotic parts below:			
CasE	*CasE / EcoCas6e*	Endoribonuclease of the Cas6 family. Cuts RNA at its recognition site, and releases the 5' end. Binds the 3' end irreversibly. Endogeneous to *E. coli*	DiAndreth et al. (2022)
Csy4		Endoribonuclease of the Cas6 family. Cuts RNA at its recognition site, and releases the 5' end. Binds the 3' end irreversibly. Endogeneous to *Pectobacterium atrosepticum*.	DiAndreth et al. (2022)

Table 13 continued from previous page

Abbreviation	Full name	Description	Source
pguCas13b		Endoribonuclease of the Cas13 family. Cuts RNA at its recognition site, and releases the 5' end. Binds the 3' end irreversibly. Endogeneous to *Porphyromonas gulae*.	DiAndreth et al. (2022)
CasE.rec	*CasE* recognition site		DiAndreth et al. (2022)
CasE.rec02	*CasE* recognition site	CasE recognition site engineered to have a different affinity to the CasE endoribonuclease than the wildtype sequence.	Section 3.2
CasE.rec03	*CasE* recognition site	See above.	Section 3.2
CasE.rec04	*CasE* recognition site	See above.	Section 3.2
CasE.rec05	*CasE* recognition site	See above.	Section 3.2
CasE.rec06	*CasE* recognition site	See above.	Section 3.2
CasE.rec07	*CasE* recognition site	See above.	Section 3.2
CasE.rec08	*CasE* recognition site	See above.	Section 3.2
CasE.rec09	*CasE* recognition site	See above.	Section 3.2
CasE.rec10	*CasE* recognition site	See above.	Section 3.2
Csy4.rec	*Csy4* recognition site		DiAndreth et al. (2022)
pguCas13b.rec	*pguCas13b* recognition site		DiAndreth et al. (2022)
eYFP	yellow fluorescent protein	Fluorescent protein. Excitation: 513 nm. Emission: 527 nm.	FPbaseID: 8DNLG

Table 13 continued from previous page

Abbreviation	Full name	Description	Source
eYFPG5A	yellow fluorescent protein	Fluorescent protein. Excitation: 513 nm. Emission: 527 nm. This protein was engineered to include a second PAM site for prime editing. It is sequence identical to eYFP, apart from a single base substitution resulting in an amino acid change from Gly to Ala at position 5.	Section 3.5
iRFP	*iRFP713*	Fluorescent protein. Excitation: 690 nm. Emission 713 nm. Requires the cofactor biliverdin, and can get dim (due to lacking cofactor) at very high expression levels.	FPbaseID: DGXFA
tagBFP		Fluorescent protein. Excitation: 402 nm. Emission: 457 nm.	FPbaseID: BFJKS
eBFP2		Fluorescent protein. Excitation: 383 nm. Emission: 448 nm.	FPbaseID: DVMQ7
mKate2		Fluorescent protein. Excitation: 588 nm. Emission: 633 nm.	FPbaseID: DBBO8

5.20 Construction Primers

Most plasmids shown in Section 5.21 were constructed using MoClo, and no primers were used for construction (see Section 5.8). The plasmids prefaced pTuring were mainly constructed by Jure Tica, Tong Zhu and Lucy Chen, with some modifications by me. The starting sequence was ordered from Twist, and the design based on parts shown in Section 5.19. Modifications were made using the primers shown in Table 14. Special thanks to Jure Tica for providing the primers used in construction in which I was not involved.

Table 14: List of primes used for cloning of the Turing (pTuring) plasmids. Primers designed by Jure Tica, Tong Zhu, Lucy Chen and Georg Wachter.

Name	Sequence	Description
LC1_fw	`AACAAAACCTCTAAACTACTGTGGCACT` `GAATCAAAAAATTATAAACCATGATCAGATC` `TAGAGAAAGAGGAGAAATACTAGATGGTGAGC`	Cloning of p3B5B promoter into pET-nodeC plasmid - forward primer. Lower case = p3B5B sequence, bold = plasmid complementary region
LC2_rv	`CGATATTTCAATAATTTGTTCGATAAT` `CGAACAAAAGGTACCCAGCCAACGATCG`	Cloning of p3B5B promoter into pET-nodeC plasmid - reverse primer. bold = plasmid complementary region

Table 14 continued from previous page

Name	Sequence	Description
LC3_fw	CACCCAAGgagctcGTTGAG	Cloning of aroZ from k. pnuemonae - forward primer amplifies pCOLA backbone
LC4_rv	AAATgaattcGGTATGGAGAAACAGTAGAG	Cloning of aroZ from k. pnuemonae - reverse primer amplifies pCOLA backbone at pBAD
LC5_fw	CGCAACTCTCTACTGTTTCTCCATACCGAATTCATTTA AAGAGGGGAAATACTAGATGCTGCGCTCTATCGCCACCG	Cloning of aroZ from k. pnuemonae - forward primer amplifies RBS-aroZ with overlap at PBAD
LC6_rv	GGTAACTCAACGAGCTCCTTGGGTGGCGAAAAAACCCCG CCGAAGCGGGGTTTTTTGCGCTAACAATACTGCATCGCCGCCAG	Cloning of aroZ from k. pnuemonae - reverse primer amplifies aroZ-TT with pCOLA overlap
LC7_fw	GAGTGGTGTATTTATTTATATTGTTTGCTCCGTTACC GTTATTAACTCTAGAGAAAGAGGAGAAATACTAGATGG	Cloning of pSalTTC promoter into pET-nodeC plasmid - forward primer
LC8_rv	GATATATAATAAATCATCAACATGAATATT GCGCCCGGCCGGGCACCAGCAATAACCCAAG CGAGGCCCCGGTACCCAGCCAACGATC	Cloning of pSalTTC promoter into pET-nodeC plasmid - reverse primer
LC9_fw	GAGATTTTGAGACACAAGGTC	Gibson assembly- Amplification of backbone with 3' homology to p23101 and 5' homology to nahRAM
LC10_rv	CAGCCAACGATCGTTAAG	Gibson assembly - Amplification of backbone with 3' homology to p23101 and 5' homology to nahRAM
LC11_fw	TCTCCCGCTTAACGATCGTTGGCTGtttacagctagct cagtcctaggtattatgctagcCATATGCGCTTACAATAGACGAAC	Gibson assembly - Amplification of p23101_pca3_pcauAM_NdeI_EcoR1 with 3' homology to nah3 and 5' homology to backbone
LC12_rv	GTTAAGTCTTTTTCTTATAGGGGGTG AATTCCATATGttacagaatgttgcgcag	Gibson assembly - Amplification of p23101_pca3_pcauAM_NdeI_EcoR1 with 3' homology to nah3 and 5' homology to backbone
LC13_fw	ACCCCCTATAAGAAAAAGAC	Gibson assembly - amplification of nah3_nahram_EcoR1_B1006 with 3' homology to TTB0053 and 5' homology to pCAUam
LC14_rv	ATTCGACCTTGTGTCTCAAAATCTCaaaaaaaa ccccgccctgtcaggggcggggttttttttt GAATTCttaatcggtaaacagatcaaac	Gibson assembly - amplification of nah3_nahram_EcoRI_B1006 with 3' homology to TTB0053 and 5' homology to pCAUam

Table 14 continued from previous page

Name	Sequence	Description
LC15_fw	TCTCCCGCTTAACGATCGTTGGCTGTTTACAGC TAGCTCAGTCCTAGGTATTATGCTAGCCATA TGCGCTTACAATAGACGAACAATAAAGG	Gibson assembly - amplification of Marionette fragment with EcoRI_B1006 with 3' homology to pca3 and 5' homology to pCC1 - use with LC16_rv
LC16_rv	CGACCTTGTGTCTCAAAATCTCAAAAAAAACCCC GCCCTGTCAGGGGCGGGGTTTTTTTTTGA ATTCTTATCAATCCGTAAACAGGTCAAAC	Gibson assembly - amplification of Marionette fragment with EcoRI_B1006 with 3' homology to NahRAM and 5' homology to pCC1 - use with LC15_fw
LC17_fw	CAACTCTCTACTGTTTCTCCATACCGAATTCATTTA AAGAGGGGAAATACTAGATGAATCACGTACCAGTAG	Cloning of pobA from c.glutamicum - forward primer amplifies RBS_pobA with overlap at PBAD
LC18_rv	GGTAACTCAACGAGCTCCTTGGGTGGCGAAAAAACCCC GCCGAAGCGGGGTTTTTTGCGTTACACTTCAAAGCGAG	Cloning of pobA from c.glutamicum - reverse primer amplifies pobA_TT with overlap with pCOLA
LC19_fw	CGCAACTCTCTACTGTTTCTCCATACCgaattcATTTaa agaggggaaatactagATGAAAATCAGTGAATTTTTACACCTGGC	Cloning of Ybts from k. pnuemoniae M5a1 - forward primer amplifies RBS_ybts with overlap at PBAD
LC20_rv	GGTAACTCAACGAGCTCCTTGGGTGGCGAAAAAACCC CGCCGAAGCGGGGTTTTTTGCGTCACGCGGGCACCTTTAG	Cloning of Ybts from k. pnuemoniae M5a1 - reverse primer amplifies Ybts_TT with overlap with pCOLA
LC21_fw	CGCAACTCTCTACTGTTTCTCCATACCgaattcATT TaaagaggggaaatactagATGAAAATCAGTGAATTTCTAC	Cloning of irp9 from yersinia entercolitica 8081 - forward primer amplifies RBS_irp9 with overlap at PBAD
LC22_rv	GGTAACTCAACGAGCTCCTTGGGTGGCGAAAAAACCC CGCCGAAGCGGGGTTTTTTGCGCTACACCATTAAATAGGGC	Cloning of irp9 from yersinia entercolitica 8081 - reverse primer amplifies irp9_TT with overlap with pCOLA
LC23_fw	CGCAACTCTCTACTGTTTCTCCATACCGAATTCA TTTAAAGAGGGGAAATACTAGATGAAAATTAGCGAATTTCTG	Cloning of irp9 from yersinia entercolitica codon optimised - forward primer amplifies RBS_irp9 with overlap at PBAD
LC24_rv	GGTAACTCAACGAGCTCCTTGGGTGGCGAAAAAACC CCGCCGAAGCGGGGTTTTTTGCGTTACACCATCAGATACG	Cloning of irp9 from yersinia entercolitica codon optimised - reverse primer amplifies irp9_TT with overlap with pCOLA

Table 14 continued from previous page

Name	Sequence	Description
LC25_fw	CAACTCTCTACTGTTTCTCCATACCGAATTCATT TAAAGAGGGGAAATACTAGCATATGCTGCGTAGCATTG	Cloning of codon optimised aroZ - forward primer amplifies RBS_aroZ with overlap at PBAD (note the codon optimised gene is flanked by RE cut sites which are apart of the annealing region).
LC26_rv	GGTAACTCAACGAGCTCCTTGGGTGGCGAAAAAACCCCGC CGAAGCGGGGTTTTTTGCGACTAGTTTAACAATACTGCATTGC	Cloning of codon optimised aroZ - reverse primer amplifies aroZ_TT with overlap with pCOLA (note the codon optimised gene is flanked by RE cut sites which are apart of the annealing region).
LC27_fw	GAGAAATACTAGATGGTGAGC	pET_px_LacOx - forward primer for introduction of LacO downstream of the promoter in pET reporters.
LC28_rv	CTCTTTCAATTGTTATCCGCTCACAAT TTCTGATCATGGTTTATAATTTTTTGATTC	pET_p3B5B_LacO1 - reverse primer introduces LacO1 downstream of the p3B5B promoter from pET_p3B5B (KLD). Use with LC27_fw
LC29_fw	CTCTAAACTACTGTGGCACTGAATCAAAAAA TTATAAACCATGATCAGAAATTGTGAGCGGATAA CAATTTCTAGAAGGAGAAATACTAGATGGTGAG	pET_p3B5B_LacO2 - forward primer for introduction of p3B5B LacO1 into pET nodeC. Use with LC30_rv in KLD.
LC30_rv	GTTTTGTTCGATATTTCAATAATTTGTTCGA AATCGAACAAAAGGTACCAATTGTTATCCGCTC ACAATTCAGCCAACGATCGTTAAG	pET_p3B5B_LacO2 - reverse primer for introduction of LacO1 p3B5B into pET nodeC. Use with LC29_fw in KLD.
LC31_rv	CTCTTTCAATTGTGAGCGCTCACAA TTTCTGATCATGGTTTATAATTTTTTGATTC	pET_p3B5B_LacOsym - reverse primer introduces LacOsym downstream of the p3B5B promoter from pET_p3B5B (KLD). Use with LC27_fw
LC32_rv	GTTTTGTTCGATATTTCAATAATTTGTTC GATAATCGAACAAAAGGTACCAATTGTGAGCG CTCACAATTCAGCCAACGATCGTTAAG	pET_p3B5B_LacOsym2 - forward primer for introduction of p3B5B LacOsym into pET nodeC. Use with LC33_fw in KLD.
LC33_fw	CTCTAAACTACTGTGGCACTGAATCAAAAA ATTATAAACCATGATCAGAAATTGTGAGCGCTCA CAATTTCTAGAAGGAGAAATACTAGATGGTGAG	pET_p3B5B_LacOsym2 - reverse primer for introduction of LacOsym p3B5B into pET nodeC. Use with LC32_rv in KLD.

Table 14 continued from previous page

Name	Sequence	Description
LC34_rv	CTAGTATTTCCCCTCTTTCTCTAG	Cloning of pCC1_aroY_pcaUAM_nahRAM - reverse primer for backbone amplification (Gibson)
LC35_fw	CGCACCAAGACAGGTTTG	Cloning of pCC1_aroY_pcaUAM_nahRAM - forward primer for backbone amplification (Gibson)
LC36_fw	GTTTAATCTAGAGAAAGAGGGGAAATACT AGATGACCGCACCGATTCAGG	Cloning of pCC1_aroY_pcaUAM_nahRAM - forward primer for amplification of aroY (Gibson) (has homology to riboJ)
LC37_rv	CTGTAAAATTCGACCCTCAAAATCTCGGGAGACC AGAAACAAAAAAAGGCCGCGTTAGCGGCCTTCAATAA TTGGTCTAGATTACTGACCGTGGCCGTTTTC	Cloning of pCC1_aroY_pcaUAM_nahRAM - reverse primer for amplification of aroY (Gibson). Introduces Terminator L3S3P22 and has homology to pca3 RBS
LC38_fw	GAGATTTTGAGGGTCGAATTTTACAGCTAGCTCAGTCC TAGGTATTATGCTAGCCATATGCGCTTACAATAGACGAAC	Cloning of pCC1_aroY_pcaUAM_nahRAM - forward primer for amplification of pcaUAM_nahRAM (Gibson). Introduces the constituitive promoter p23101 and has homolgy to LC38_fw
LC39_rv	CAATGGACAAACCTGTCTTGGTGCGAAAAAAAACCCCGCCCTGT CAGGGGCGGGGTTTTTTTTTGAATTCTTATCAATCCGTAAACAGGTC	Cloning of pCC1_aroY_pcaUAM_nahRAM - reverse primer for amplification of pcaUAM_nahRAM (Gibson). Introduces terminator B1006 and has homology to pCC1
LC40_rv	GATATTTCAATAATTTGTTCGATAATCG AACAAAAGTCGACCAGCCAACGATCGTTAAGC	Cloning of p3B5B promoter into pCDF_nodeB - reverse primer maintaining one cI* site (KLD)
LC41_fw	GAACAAAACCTCTAAACTACTGTGGCACTGAATCAAAAAA TTATAAACCATGATCAGATATCGGCGCCGCCGATAAGC	Cloning of p3B5B promoter into pCDF_nodeB - forward primer maintaining one cI* site (KLD)
Blue_1_rv	GTACCTAGGACTGAGCTAGCCGTC AAGCTATGGCATAGCAAAGTGTGACGC	Cloning of pCOLA_BFP - reverse primer for amplification of pCOLA backbone. Homology to pJ23100

Table 14 continued from previous page

Name	Sequence	Description
Blue_2_fw	CTGCAAACGACGAAAACTACGCTTTAGTAGCTTTACC TGCAGCGCAAAAAACCCCGCTTCGGCGGGGTTTTTT CGCTCTAGATGCGCACCACCTGCAGG	Cloning of pCOLA_BFP - forward primer for amplification of pCOLA backbone.Introduces LVA tag and B1002 terminator and XbaI site
Blue_3_fw	TTGACGGCTAGCTCAGTCCTAGGTACAGTGCTAGCGAA TTCCCAAAGAGGGGAAAAAGGAGCTCATGTCCGAGTTGATCAAAGAG	Clonining of pCOLA_BFP - forward primer for BFP amplificiation. Introduces strong promoter 23100 and RBS B0064.
Blue_4_rv	CTAAAGCGTAGTTTTCGTCGTTTGCA GCCTCAACATTAAGCTTGTGGCCCAGTTTG	Cloning of pCOLA_BFP - reverse primer for amplification of BFP. Homology to LVA.
edit_LC34	GTAGAGGCCCCATTTCGTCCTTTTGGACTCATCAGGGGTGGTACAC ACCACCCTATGGGGCTATTGGATCCAATGGTACCTAGG	Correction - cloning of pCC1_aroY_pcaUAM_nahRAM. Reverse primer for backbone amplification use with LC35. Binds to pVan promtoer and contain half of the elvJ insulator
edit_LC36	CAAAAGGACGAAATGGGGCCTCTACAAATAA TTTTGTTTAATCTAGAGTCACACAGGAAA GACTAGTATGACCGCACCGATTCAG	Correction - cloning of pCC1_aroY_pcaUAM_nahRAM. Forward primer for aroY amplification use with edit_LC37. Contains half of the elvJ insulator and introduces a 5' SpeI site
edit_LC37	GCTGTAAAATTCGACCCTCAAAATCTCGGGAG ACCAGAAACAAAAAAAGGCCGCGTTAGCGGCC TTCAATAATTGGACTAGTTTATTTTGCGCTACCCTGG	Correction - cloning of pCC1_aroY_pcaUAM_nahRAM. Reverse primer for aroY amplification use with edit_LC37. Contains TT_L3S3P22 and introduces a 3' SpeI site
edit_LC38	GATTTTGAGGGTCGAATTTTACAGCTAGCTCAGTCCT AGGTATTATGCTAGCTCATGACGCTTACAATAGACGAAC	Correction - cloning of pCC1_aroY_pcaUAM_nahRAM. Forward primer for p23101_pca3_etc introduction. Removes NdeI site.
edit_LC40	CCGATATCTGATCATGGTTTATAATTTTTTGATTCAGTGCCA CAGTAGTTTAGAGGTTTTGTTCGATATTTCAATAATTT GTTCGATAATCGAACAAAAGTCGACCAGCCAACGATC	Corrected - reverse primer for cloning of pCDF_p3B5B_ElvJ_nodeB. Replaces riboJ with ElvJ insulator http://dx.doi.org/10.1126/science.aac7341.

Table 14 continued from previous page

Name	Sequence	Description
edit_LC41	CGCCGCCGATAAGCCCCATAGGGTGGTGTGTACCAC CCCTGATGAGTCCAAAAGGACGAAATGGGGCCTCTACAAATA ATTTTGTTTAAGGATCCAAAGAGGGGAAAGAATTCATGttcg	Corrected - forward primer for cloning of pCDF_p3B5B_ElvJ_nodeB. Replaces riboJ with ElvJ insulator http://dx.doi.org/10.1126/science.aac7341.
LC42_fw	CAAAAGGACGAAATGGGGCCTCTACAAATAATTTT GTTTAATCTAGAGTCACACAGGAAAGCATATG ATGCAGAACAGTACCAGCGC	pCC1_nahG cloning - amplification of nahG_opt synthesised from twist. Introduces a NdeI site. Use with edit34, 35, edit38, 39 fragments for gibson assembly
LC43_rv	GCTGTAAAATTCGACCCTCAAAATCTCGGGAGACCA GAAACAAAAAAAGGCCGCGTTAGCGGCCTTCAATAAT TGGACTAGTTTAGGCGGCTTGGGCACG	pCC1_nahG cloning - amplification of nahG_opt synthesised from twist. Introduces a SpeI site. Use with edit34, 35, edit38, 39 fragments for gibson assembly
LC44_rv	GATATATAATAAATCATCAACATGAATATTGCGCCC GGCCGGGCACCAGCAATAACCCAAGCGAGGCCCC GTCGACCAGCCAACGATCGTTAAG	pCDF_nodeB_pSalTTC - KLD
LC45_fw	GAGTGGTGTATTTATTTATATTGTTTGCTCC GTTACCGTTATTAACTATCGGCGCCGCCGATAAG	pCDF_nodeB_pSalTTC - KLD
LC46_ rv	CCCCGAATTCTTAAACAAAATTATTTGTAGAGGCTG	pCOLA_NodeA_PCA - Cloning step 1: amplifies backbone with a 3' EcoRI site.
LC47_fw	CCCCCTCGAGGCTGCAAACGACGAAAAC	pCOLA_NodeA_PCA - Cloning step 1: amplifies backbone with a 5' XhoI site.
LC48_fw	CCCCGAATTCTCACACAGGAAAGACTCAGATGCTGCGCTCTATCG	pCOLA_NodeA_PCA - Cloning step 1: amplifies aroZ with 5' EcoRI site and introduces the RBS B0032.
LC49_rv	CCCCCTCGAGCTAACAATACTGCATCGCC	pCOLA_NodeA_PCA - Cloning step 1: amplifies aroZ with 3' XhoI site.
LC50_fw	CCCCGAATTCTCACACAGGAAAGACTCA GATGAAAATCAGTGAATTTTTACACCTGGC	pCOLA_NodeA_PCA - Cloning step 1: amplifies ybts with 5' EcoRI site and introduces the RBS B0032.
LC51_rv	CCCCCTCGAGTCACGCGGGCACCTTTAG	pCOLA_NodeA_PCA - Cloning step 1: amplifies ybts with 3' XhoI site.
LC52_fw	GAGTGGTGTATTTATTTATATTGTTTGCTCC GTTACCGTTATTAACAATTGTGAGCGGATAACAA TTGGATCCAGCTGTCACCGGATGTGC	Forward primer to introduce PSalTTC promoter with single downstream LacO into pCOLA_node A_ybts
LC53_rv	GATATATAATAAATCATCAACATGAATATTGCGCCCGGCCG GGCACCAGCAATAACCCAAGCGAGGCCCCGGTACCCAGCCAACGATCG	Reverse primer to introduce PSalTTC promoter with single downstream LacO into pCOLA_node A_ybts

Table 14 continued from previous page

Name	Sequence	Description
LC54_rv	CTCTAGACCATGGTTATCA	pCDF_nodeB_p3B5B_dcI - reverse primer to remove WT cI (KLD)
LC55_fw	CTGATAATCTAGAGAAAGAG	pCDF_nodeB_p3B5B_dcI - forward primer to remove WT cI (KLD)
LC56_rv	GGCTTCTGATCATGGTTTATAATTTTTTG	pCDF_nodeB_p3B5B_dcIO* - reverse primer to remove clO* operator from node B (KLD)
LC57_fw	CCATAGGGTGGTGTGTAC	pCDF_nodeB_p3B5B_dcIO* - reverse primer to remove clO* operator from node B (KLD)
LC58_rv	GCGCCGATATCTGATCATGGTTTATAATTTTTTG	pET_p3B5B_cIO* - reverse primer to introduce clO* operator into RFP reporter (KLD)
LC59_fw	CGCCGATATCTAGAGAAAGAGGAGAAATAC	pET_p3B5B_cIO* - forward primer to introduce clO* operator into RFP reporter (KLD)
LC60_rv	CACAGTAGTTTAGAGGTTTTGTTCGATATTTCAATAA TTTGTTCGATAATCGAACAAAAGGTACCCAGCCAACGATCG	Forward primer to introduce P3B5B promoter with single downstream LacO into pCOLA_node A_aroZ
LC61_fw	GCACTGAATCAAAAAATTATAAACCATGATCAGAAAT TGTGAGCGGATAACAATTGGATCCAGCTGTCACCGGATGTGC	Reverse primer to introduce P3B5B promoter with single downstream LacO into pCOLA_node A_aroZ
LC63_rv	CACAGTAGTTTAGAGGTTTTGTTCGATATTTCAATAAT TTGTTCGATAATCGAACAAAAGTCGACAATTGTGAGCGCTC	Reverse primer to introduce P3B5B promoter with upstream LacOsym and single downstream LacO into pCOLA_node A_aroZ, use with LC61_fw
LC64	CTGCAAACGACGAAAACTACGC	KLD reaction to fix LVA degradation tag by replacing stop codon of coding sequence
LC65	CACAATACTGCATCGCCGC	KLD reaction to fix LVA degradation tag by replacing stop codon of coding sequence
LC66	CCGCGGGCACCTTTAGATAG	KLD reaction to fix LVA degradation tag by replacing stop codon of coding sequence
LC67	CTATTACTCATGTGCCTGAATGATAAC	KLD reaction for introducing two stop codons at position 10 of cinI in node B. Use with ztB2_in_fw.

Table 14 continued from previous page

Name	Sequence	Description
LC68_rv	GTCCTTTTGGACTCATC	KLD reaction for introducing weaker RBS infront of cinI
LC69_fw	GAAATGGGGCCTCTACAAATAATTTTGTTTAAGGA TCCTCACACAGGAAACCGAATTCATGTTCGTTATCATTC	KLD reaction for introducing weak RBS (0.07%) B0031 infront of cinI. Use with LC68
LC70_fw	GAAATGGGGCCTCTACAAATAATTTTGTTTAAGGA TCCTCACACAGGACGAATTCATGTTCGTTATCATTC	KLD reaction for introducing weakest RBS (0.01%) B0033 infront of cinI. Use with LC68
GW001_fw	TGATAATAGAGGCATCAAATAAAACG	pET-C amplification of backbone. includes stop codon
GW001_rv	GCTAGCACTGTACCTAGG	bb amplification (pET-C)
GW002_fw	gctcagtcctaggtacagtgctagc ATGAAAATCAGTGAATTTTTACACCTGGC	YBTS amplification
GW002_rv	gttttatttgatgcctctattatcaCGCGGGCACCTTTAGATAGG	YBTS amplification
GW003_fw	gctcagtcctaggtacagtgctagc ATGTCTAGATTAGATAAAAGTAAAGTG	tetR amplification
GW003_rv	gttttatttgatgcctctattatcaGGACCCACTTTCACATTTAAG	tetR amplification
GW004_fw	gctcagtcctaggtacagtgctagcATGTTCGTTATCATTCAGGCAC	cinI amplification
GW004_rv	gttttatttgatgcctctattatcaACTAGTTGCTGCCATCTCC	cinI amplification
GW005_fw	gctcagtcctaggtacagtgctagc ATGAAACCAGTAACGTTATACGATGTCG	LacI amplification
GW005_rv	gttttatttgatgcctctattatcaCTGCCCGCTTTCCAGTCG	LacI amplification
GW006_fw	gctcagtcctaggtacagtgctagcATGAGCACAAAAAAGAAACC	cI and cI* amplification
GW006_rv	gttttatttgatgcctctattatcaGCCAAACGTCTCTTCAGG	cI amplification
GW007_rv	gttttatttgatgcctctattatcaGCCAAACGTTTCTTCAGG	cI* amplification
GW040_fw	CTACAAATAATTTTGTTTAATGATAATCTAGAGAAAGAGGAGAAATAC	mCherry amplification to enter pCOLA
GW040_rv	tatttgatgcctggccatggAACGCTAGCTGCGTAGTTTTC	mCherry amplification to enter pCOLA
GW041_fw	ccatggccaggcatcaaataaaacg	pCOLA binding
GW041_rv	TTAAACAAAATTATTTGTAGAGGTGGTTTCGCTCTGCCTTG AGCTCATCAGGTGATCGAGTTTCAGATCACGACCACTTT ATCAGGCGACCAGAGCATAGTTTTC	pCOLA binding
GW042_fw	CTACAAATAATTTTGTTTAATGATAATCTAGAGAAAGAGGAGAAATAC	mCherry amplification to enter pCDF
GW042_rv	TACCGAGcaataattggttaAACGCTAGCTGCGTAGTTTTC	mCherry amplification to enter pCDF
GW043_fw	taaccaattattgCTCGGTACC	pCDF backbone amplification
GW043_rv	TTAAACAAAATTATTTGTAGAGGTGGTTTCGCTC TGCCTTGAGCTCATCAGGTGATCGAGTTTCAGATCAC GACCACTTCAACTAGTTCAagctactaaagcg	pCDF backbone amplification
GW044_fw	AGCAACCCAGTCAGCTCC	pCDF backbone amplification

Table 14 continued from previous page

Name	Sequence	Description
GW044_rv	TTAAACAAAATTATTTGTAGAGGCCCC	pCDF backbone amplification
GW045_fw	CTACAAATAATTTTGTTTAATGATAATCTAGAGAAAGAGGAGAAATAC	mCherry amplification for Prpa pCDF test plasmid
GW045_rv	aaggagctgactgggttgCTGGACCAAAA CGAAAAAAGGCCCCCCTTTCGGGAGGCCTCTTTTCTGGAATTT GGTACCGAGcaataattggttaAACGCTAGCTGCGTAGTTTTC	mCherry amplification for Prpa pCDF test plasmid
GW046_fw	AGCAACCCAGTCAGCTCC	pCDF backbone amplification for cinI and mCherry plasmid
GW046_rv	AGTTTCAGATCACGACCACTTCAagctactaaagcgtagttttcgtcg	pCDF backbone amplification for cinI and mCherry plasmid
GW047_fw	AGTGGTCGTGATCTGAAACTCGATCACCTGATGAGCT CAAGGCAGAGCGAAACCACCTCTACAAATAAT TTTGTTTAATGATAATCTAGAGAAAGAGGAGAAATAC	mCherry amplification for cinI and mCherry plasmid
GW048_fw	AAAACTACGCAgctAGCgttACTAGTTGATAACCAATTATTGCTCGG	pCDF backbone amplification for pC-cinImut mCherry plasmid
GW048_rv	ggagcaagcgaatcgtagctGTTacgTTTgtaagggccgatgacagg	pCDF backbone amplification for pC-cinImut mCherry plasmid
GW049_fw	agctacgattcgcttgc	cinI amplification for pC-cinImut mCherry plasmid
GW049_rv	AGTTTCAGATCACGACCACTTCAagctactaaagcgtag	cinI amplification for pC-cinImut mCherry plasmid
GW050_fw	AGTGGTCGTGATCTGAAACTCGATCACC TGATGAGCTCAAGGCAGAGCGAAACCACCTCTACAAATAA TTTTGTTTAATGATAATCTAGAGAAAGAGGAGAAATAC	mCherry amplification for pC-cinImut mCherry plasmid
GW050_rv	AACGCTAGCTGCGTAGTTTTC	mCherry amplification for pC-cinImut mCherry plasmid
GW051_fw	AAAcgtAACagctacgattcgcttgc	pCDF cinI mut plasmid via KLD from pCDF-nodeB
GW051_rv	gtaagggccgatgacagg	pCDF cinI mut plasmid via KLD from pCDF-nodeB

5.21 Plasmids Used

This section contains a complete list of genetic parts and plasmids used to produce the data in this thesis. Plasmids prefixed with pTuring were assembled by Jure Tica, Tong Zhu, and myself in the Isalan lab at Imperial College London. These plasmids were all assembled using Gibson assembly (Section 5.7). Any scarless assembly method may be used, but a list of assembly primers used to assemble the parts listed in Section 5.19 via Gibson assembly is provided in Section 5.20. Enumerated plasmids (e.g. pGW0042) were assembled using

MoClo (Section 5.8) in the Weiss lab at MIT, and are attributed according to the initials in the prefix:

pGW: Georg K. A. Wachter.

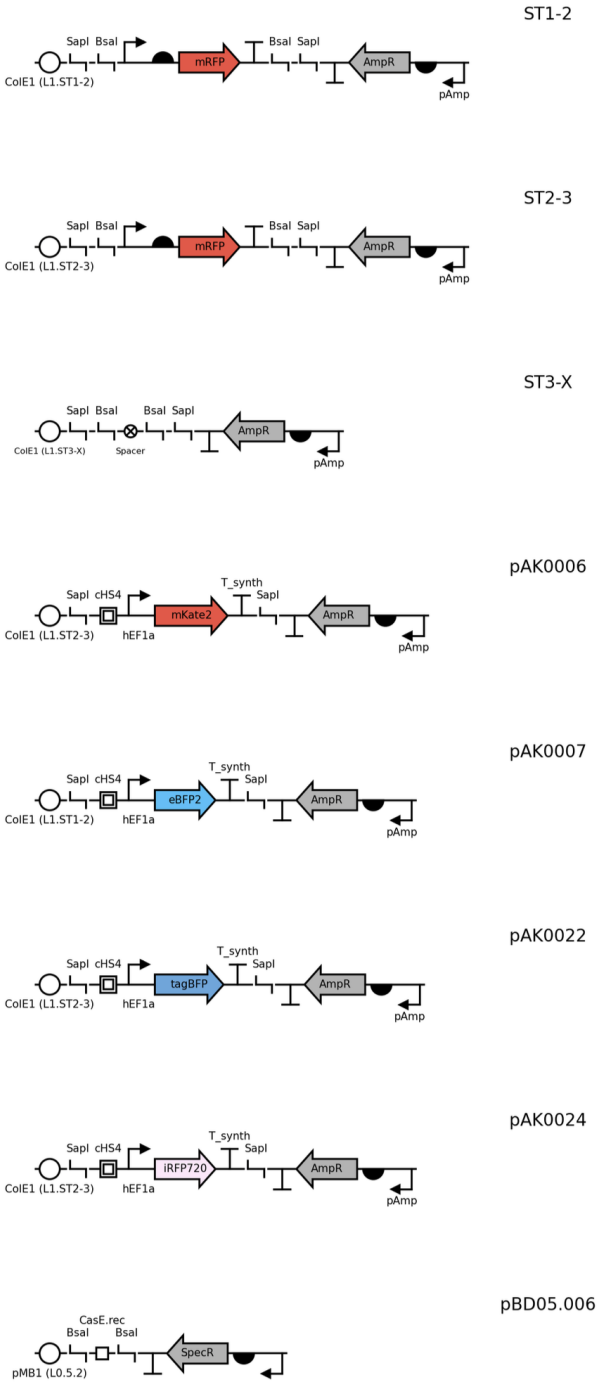
pSEP: Sarah E. Pertsemlidis.

pNW: Noreen Wauford.

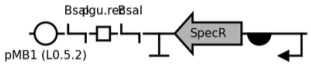
pBD: Bre DiAndreth.

pAK: Andrew Kane.

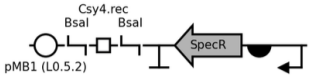
Figure 71: The following is a complete list of plasmids used in SBOL format.



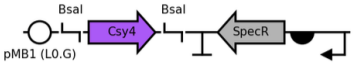
pBD05.019



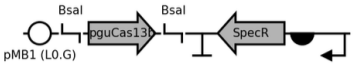
pBD05.05



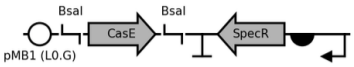
pBD06.007



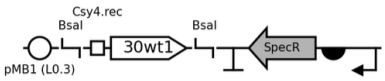
pBD06.021



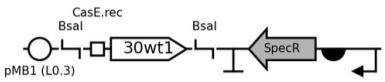
pBD06.09



pBD07.011



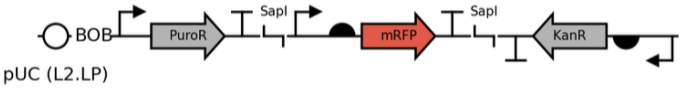
pBD07.015



pBD15.145



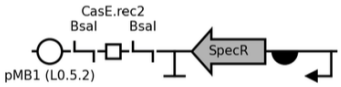
pBD23.XcR



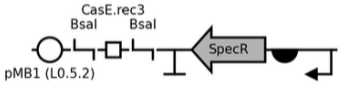
pBD24.XcR



pGW0001



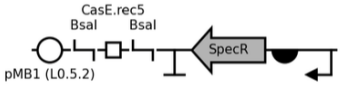
pGW0002



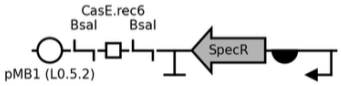
pGW0003



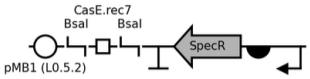
pGW0004



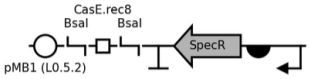
pGW0005



pGW0006



pGW0007



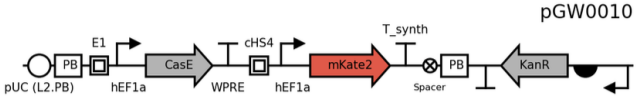
pGW0008



pGW0009



pGW0010



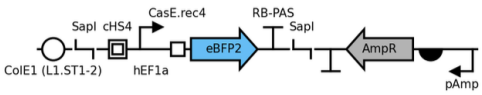
pGW0011



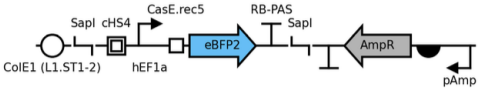
pGW0012



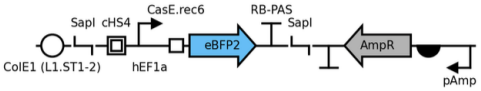
pGW0013



pGW0014



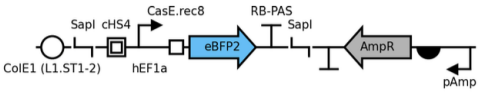
pGW0015



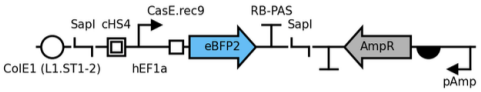
pGW0016



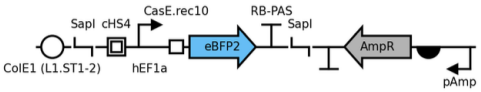
pGW0017



pGW0018



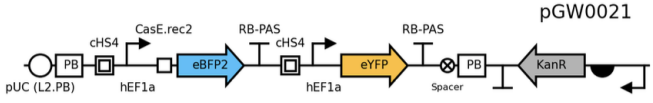
pGW0019



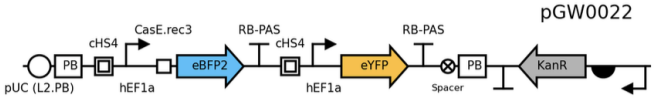
pGW0020



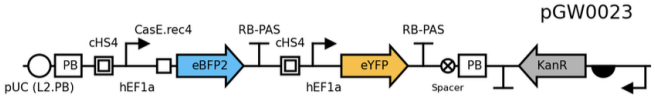
pGW0021



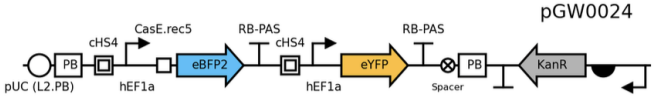
pGW0022



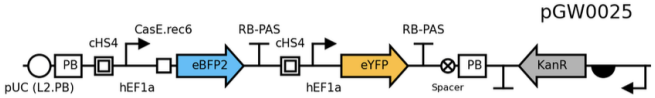
pGW0023



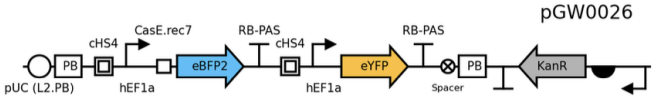
pGW0024



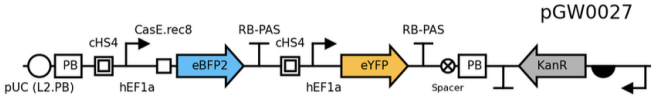
pGW0025



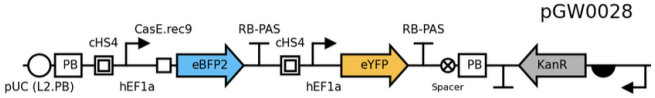
pGW0026



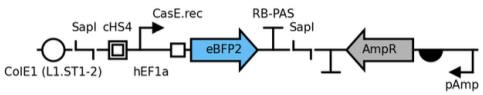
pGW0027

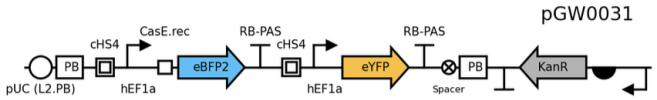


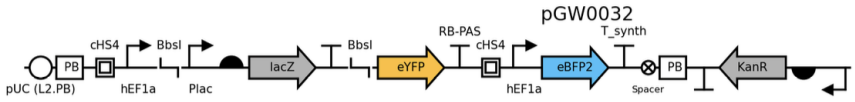
pGW0028

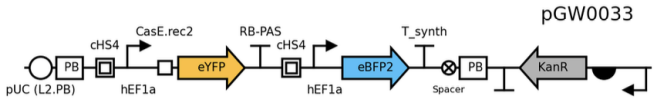


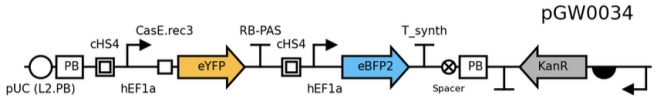
pGW0030

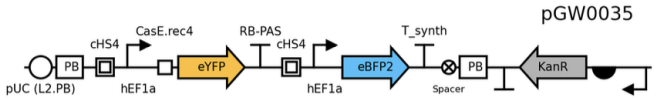


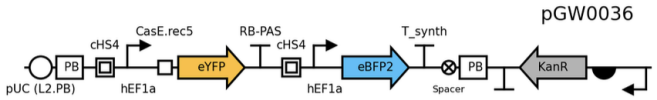

pGW0031
cHS4
CasE.rec
RB-PAS
cHS4
RB-PAS
PB
eBFP2
eYFP
PB
KanR
pUC (L2.PB)
hEF1a
hEF1a
Spacer

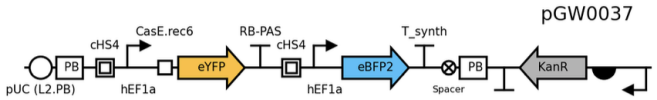

pGW0032
cHS4
BbsI
RB-PAS
cHS4
T_synth
PB
lacZ
BbsI
eYFP
eBFP2
PB
KanR
pUC (L2.PB)
hEF1a
Plac
hEF1a
Spacer

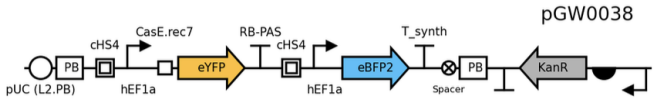

pGW0033
cHS4
CasE.rec2
RB-PAS
cHS4
T_synth
PB
eYFP
eBFP2
PB
KanR
pUC (L2.PB)
hEF1a
hEF1a
Spacer


pGW0034
cHS4
CasE.rec3
RB-PAS
cHS4
T_synth
PB
eYFP
eBFP2
PB
KanR
pUC (L2.PB)
hEF1a
hEF1a
Spacer

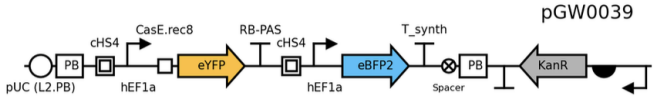

pGW0035
cHS4
CasE.rec4
RB-PAS
cHS4
T_synth
PB
eYFP
eBFP2
PB
KanR
pUC (L2.PB)
hEF1a
hEF1a
Spacer


pGW0036
cHS4
CasE.rec5
RB-PAS
cHS4
T_synth
PB
eYFP
eBFP2
PB
KanR
pUC (L2.PB)
hEF1a
hEF1a
Spacer

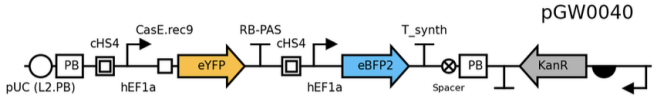

pGW0037
cHS4
CasE.rec6
RB-PAS
cHS4
T_synth
PB
eYFP
eBFP2
PB
KanR
pUC (L2.PB)
hEF1a
hEF1a
Spacer


pGW0038
cHS4
CasE.rec7
RB-PAS
cHS4
T_synth
PB
eYFP
eBFP2
PB
KanR
pUC (L2.PB)
hEF1a
hEF1a
Spacer

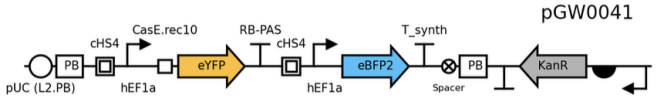
pGW0039



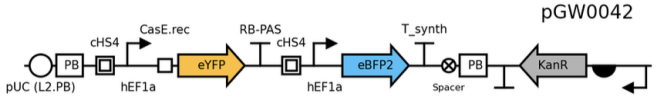
pGW0040



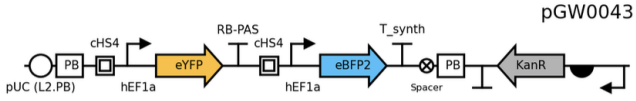
pGW0041



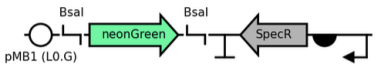
pGW0042



pGW0043



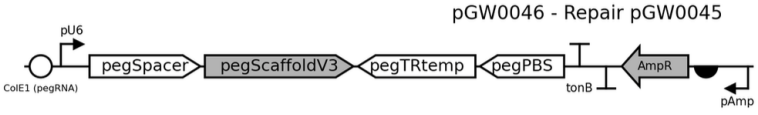
pGW0044



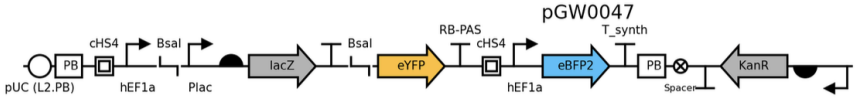
pGW0045



pGW0046 - Repair pGW0045



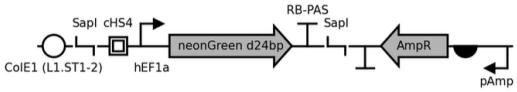
pGW0047



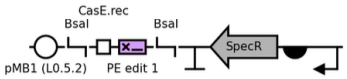
pGW0048



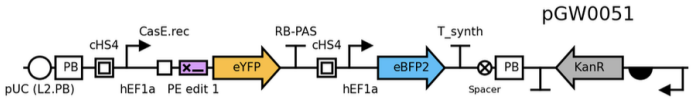
pGW0049



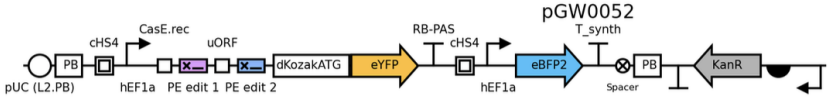
pGW0050



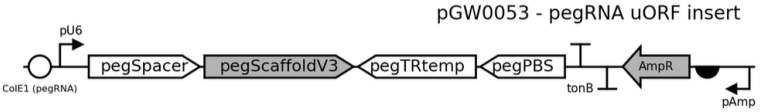
pGW0051



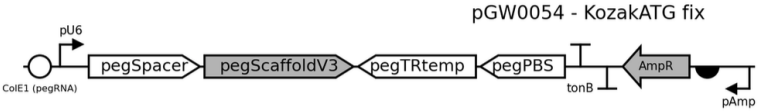
pGW0052



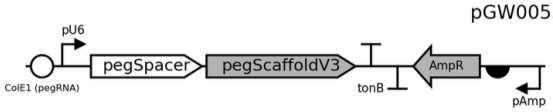
pGW0053 - pegRNA uORF insert



pGW0054 - KozakATG fix



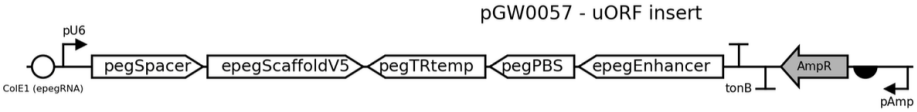
pGW0055 - Enhancer Cut



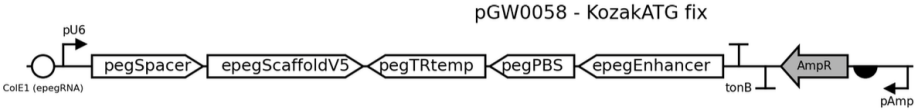
pGW0056



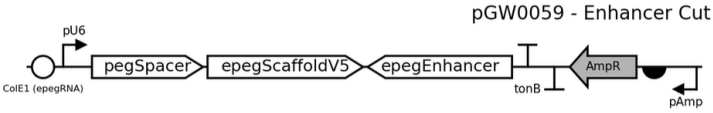
pGW0057 - uORF insert



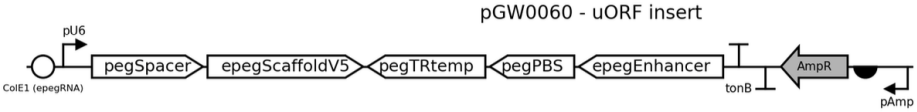
pGW0058 - KozakATG fix



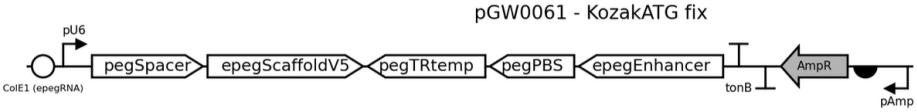
pGW0059 - Enhancer Cut



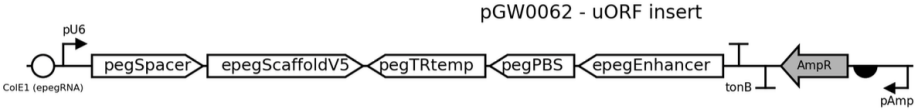
pGW0060 - uORF insert



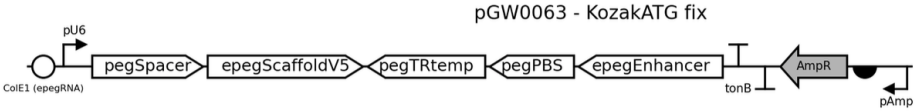
pGW0061 - KozakATG fix



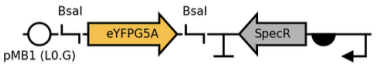
pGW0062 - uORF insert



pGW0063 - KozakATG fix



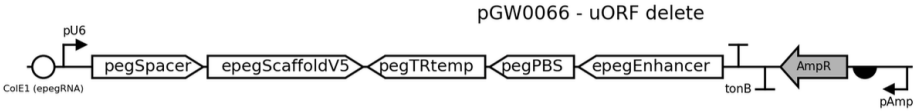
pGW0064



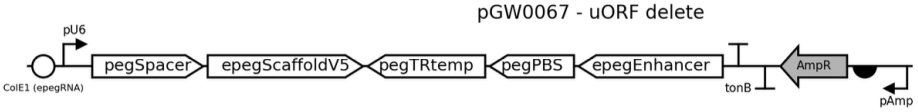
pGW0065



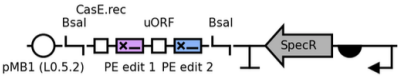
pGW0066 - uORF delete



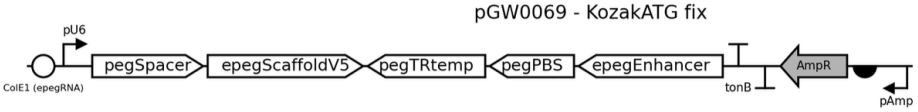
pGW0067 - uORF delete



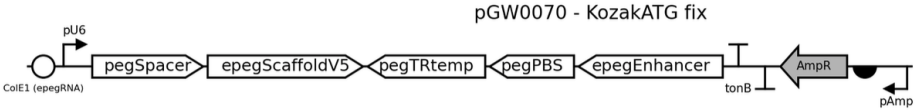
pGW0068



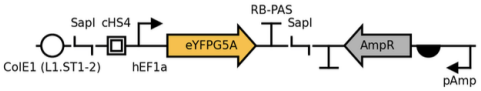
pGW0069 - KozakATG fix



pGW0070 - KozakATG fix



pGW0071



pGW0072



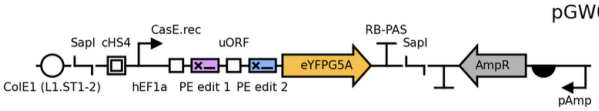
pGW0073



pGW0074



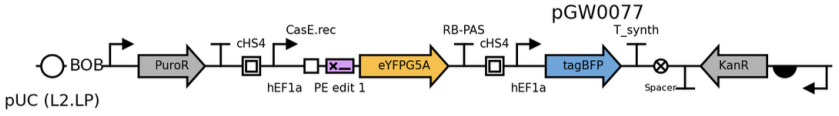
pGW0075



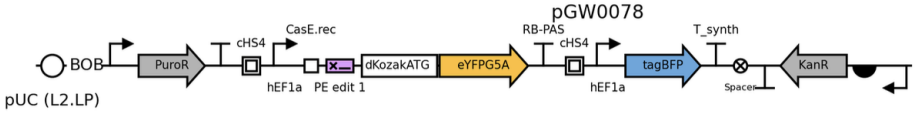
pGW0076



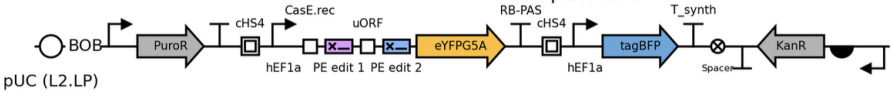
pGW0077



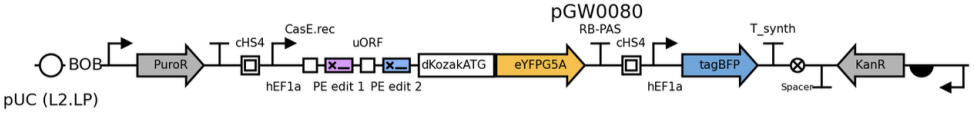
pGW0078



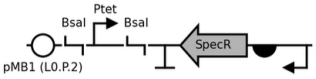
pGW0079



pGW0080



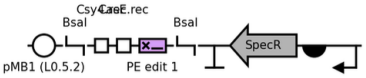
pGW0081



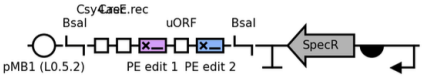
pGW0082



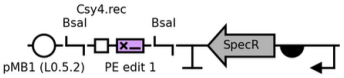
pGW0083



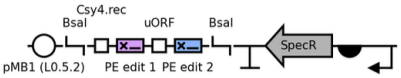
pGW0084



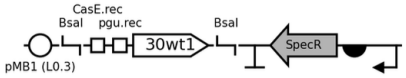
pGW0085



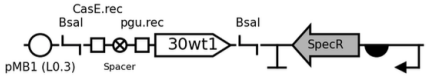
pGW0086



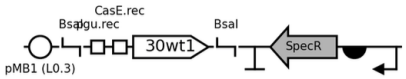
pGW0087



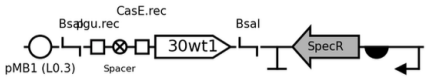
pGW0088



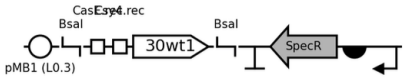
pGW0089



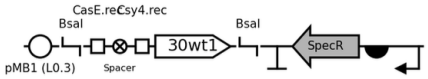
pGW0090



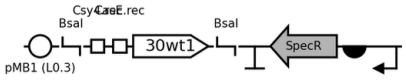
pGW0091



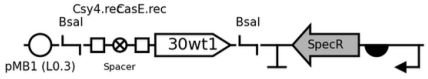
pGW0092



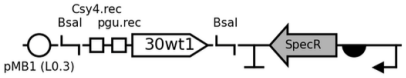
pGW0093



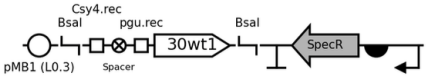
pGW0094



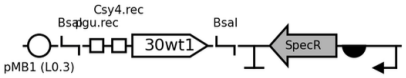
pGW0095



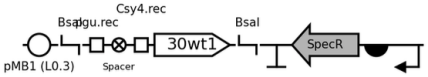
pGW0096



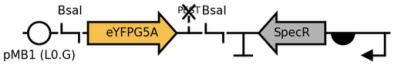
pGW0097



pGW0098



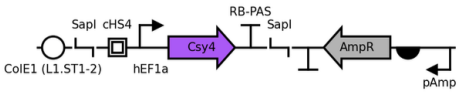
pGW0099



pGW0100



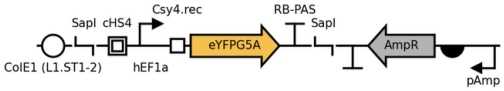
pGW0101



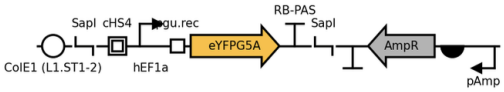
pGW0102



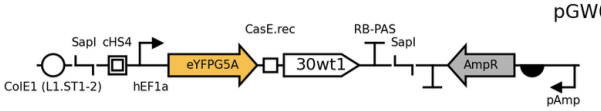
pGW0103



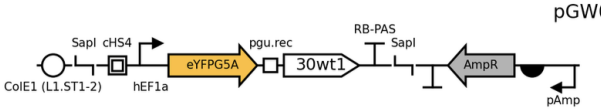
pGW0104



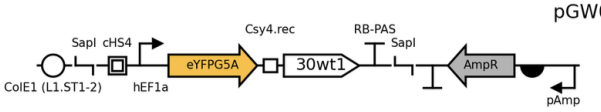
pGW0105



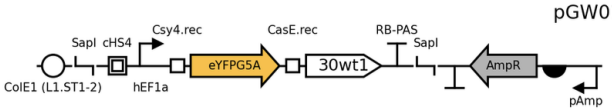
pGW0106



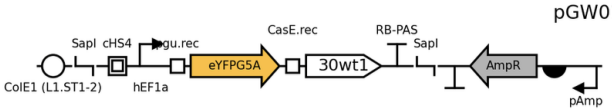
pGW0107



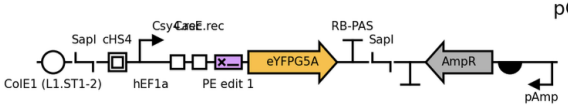
pGW0109



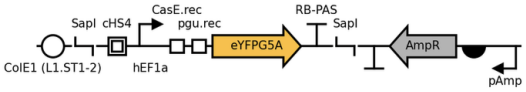
pGW0110



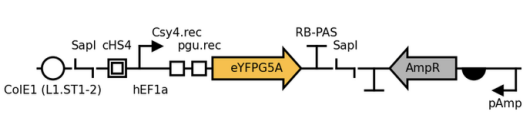
pGW0111



pGW0112



pGW0113



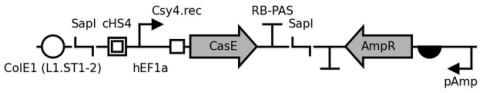
pGW0114



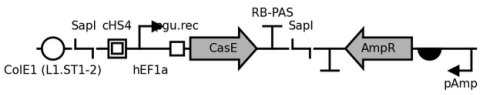
pGW0115



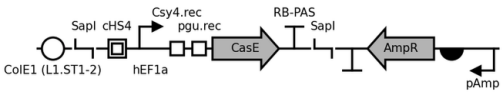
pGW0116



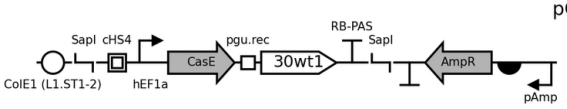
pGW0117



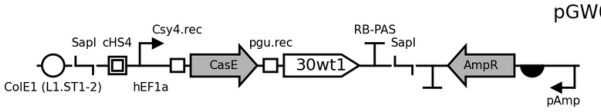
pGW0118



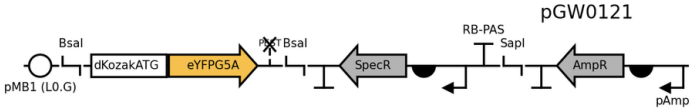
pGW0119



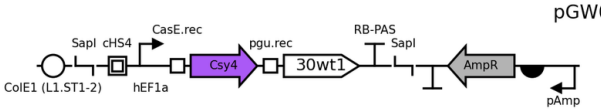
pGW0120



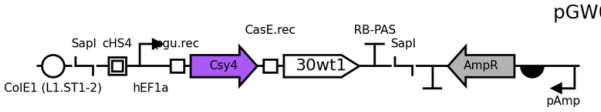
pGW0121



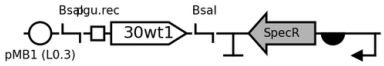
pGW0122



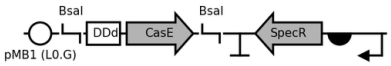
pGW0123



pNW03.013



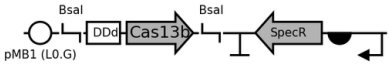
pNW06.025



pNW06.026



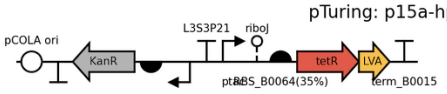
pNW06.071



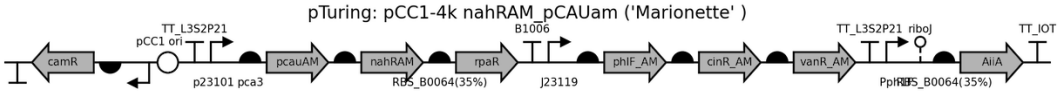
pSEP11



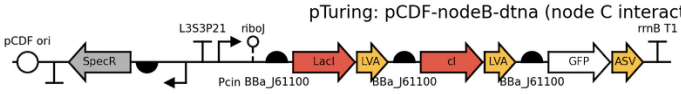
pTuring: p15a-hpv0t (Liquid culture characterisation)



pTuring: pCC1-4k nahRAM_pCAUam ('Marionette')



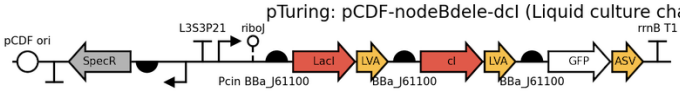
pTuring: pCDF-nodeB-dtna (node C interaction only)



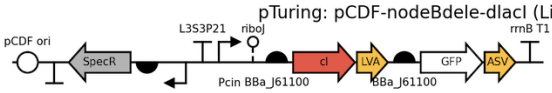
pTuring: pCDF-nodeB_SA_0.01%cinI



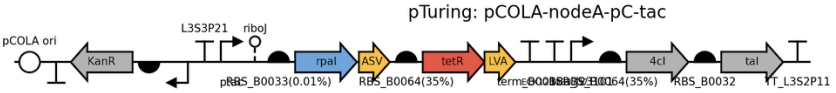
pTuring: pCDF-nodeBdele-dcI (Liquid culture characterisation)



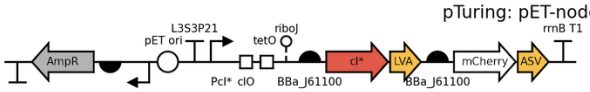
pTuring: pCDF-nodeBdele-dlacI (Liquid culture characterisation)



pTuring: pCOLA-nodeA-pC-tac



pTuring: pET-nodeC



6

Reference list

References

1. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Umar, A. M., Linus, O. U., Arshad, H., Kazaure, A. A., Gana, U. and Kiru, M. U. (2019), 'Comprehensive review of artificial neural network applications to pattern recognition', *IEEE Access* **7**, 158820–158846.
2. Al-Radhawi, M. A., Tran, A. P., Ernst, E. A., Chen, T., Voigt, C. A. and Sontag, E. D. (2020), 'Distributed implementation of boolean functions by transcriptional synthetic circuits', *ACS Synthetic Biology* **9**(8), 2172–2187.
URL: *https://doi.org/10.1021/acssynbio.0c00228*
3. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. (2008), *Molecular biology of the cell*, 5th ed. edn, Garland Science, New York.
4. Almuedo-Castillo, M., Bläßle, A., Mörsdorf, D., Marcon, L., Soh, G. H., Rogers, K. W., Schier, A. F. and Müller, P. (2018), 'Scale-invariant patterning by size-dependent inhibition of Nodal signalling', *Nature Cell Biology* **20**(9), 1032–1042.
URL: *https://doi.org/10.1038/s41556-018-0155-7*
5. Andrianantoandro, E., Basu, S., Karig, D. K. and Weiss, R. (2006), 'Synthetic biology: new engineering rules for an emerging discipline', *Molecular Systems Biology* **2**(1), 2006.0028.
URL: *https://www.embopress.org/doi/abs/10.1038/msb4100073*
6. Anzalone, A. V., Gao, X. D., Podracky, C. J., Nelson, A. T., Koblan, L. W., Raguram, A., Levy, J. M., Mercer, J. A. M. and Liu, D. R. (2021), 'Programmable deletion, replacement, integration and inversion of large DNA sequences with twin prime editing', *Nature Biotechnology* .
URL: *https://doi.org/10.1038/s41587-021-01133-w*
7. Anzalone, A. V., Randolph, P. B., Davis, J. R., Sousa, A. A., Koblan, L. W., Levy, J. M., Chen, P. J., Wilson, C., Newby, G. A., Raguram, A. and Liu, D. R. (2019), 'Search-and-replace genome editing without double-strand breaks or donor DNA', *Nature* **576**(7785), 149–157.
URL: *https://doi.org/10.1038/s41586-019-1711-4*
8. Babuska, I., Flaherty, J., Henshaw, W., Hopcroft, J., Oliger, J. and Tezduyar, T. (2012), *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, The IMA Volumes in Mathematics and its Applications, Springer New York.
URL: *https://books.google.at/books?id=JdTTBwAAQBAJ*
9. Baig, H., Fontanarrosa, P., Kulkarni, V., McLaughlin, J. A., Vaidyanathan, P., Bartley, B., Beal, J., Crowther, M., Gorochowski, T. E., Grünberg, R., Misirli, G., Scott-Brown, J., Oberortner, E., Wipat, A. and Myers, C. J. (2020), 'Synthetic biology open language (SBOL) version 3.0.0', *Journal of Integrative Bioinformatics* **-1**(ahead-of-print).
10. Baker, R., Schnell, S. and Maini, P. (2006), 'A clock and wavefront mechanism for somite formation', *Developmental Biology* **293**(1), 116–126.
URL: *https://www.sciencedirect.com/science/article/pii/S0012160606000558*
11. Banoula, M. (2023), 'What is perceptron: A beginners guide for perceptron'.
URL: *https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron*
12. Basu, S., Arnold, F. H., Gerchman, Y., Weiss, R. and Collins, C. H. (2005), 'A synthetic multicellular system for programmed pattern formation', *Nature* **434**(7037), 1130–1134.
13. Basu, S., Karig, D. and Weiss, R. (2003), 'Engineering signal processing in cells: Towards molecular

concentration band detection', *Natural computing* **2**(4), 463–478.

14. Beal, J., Overney, C., Adler, A., Yaman, F., Tiberio, L. and Samineni, M. (2019), 'TASBE flow analytics: A package for calibrated flow cytometry analysis', *ACS Synthetic Biology* **8**(7), 1524–1529.
URL: *https://doi.org/10.1021/acssynbio.8b00533*

15. Biancalani, T., Fanelli, D. and Di Patti, F. (2010), 'Stochastic turing patterns in the Brusselator model', *Phys. Rev. E* **81**, 046215.
URL: *https://link.aps.org/doi/10.1103/PhysRevE.81.046215*

16. Biancalani, T., Jafarpour, F. and Goldenfeld, N. (2017), 'Giant amplification of noise in fluctuation-induced pattern formation', *Phys. Rev. Lett.* **118**, 018101.
URL: *https://link.aps.org/doi/10.1103/PhysRevLett.118.018101*

17. Boehm, C. R., Grant, P. K. and Haseloff, J. (2018), 'Programmed hierarchical patterning of bacterial populations', *Nature Communications* **9**(1), 776.
URL: *https://doi.org/10.1038/s41467-018-03069-3*

18. Bourke, P. (1991), 'L-system user notes'.

19. Brophy, J. A. N. and Voigt, C. A. (2014), 'Principles of genetic circuit design', *Nature Methods* **11**(5), 508–520.
URL: *https://doi.org/10.1038/nmeth.2926*

20. Brödel, A. K., Jaramillo, A. and Isalan, M. (2016), 'Engineering orthogonal dual transcription factors for multi-input synthetic promoters', *Nature Communications* **7**(1).

21. Butler, T. and Goldenfeld, N. (2009), 'Robust ecological pattern formation induced by demographic noise', *Phys. Rev. E* **80**, 030902.
URL: *https://link.aps.org/doi/10.1103/PhysRevE.80.030902*

22. Butler, T. and Goldenfeld, N. (2011), 'Fluctuation-driven turing patterns', *Phys. Rev. E* **84**, 011112.
URL: *https://link.aps.org/doi/10.1103/PhysRevE.84.011112*

23. Cabrera, A., Edelstein, H. I., Glykofrydis, F., Love, K. S., Palacios, S., Tycko, J., Zhang, M., Lensch, S., Shields, C. E., Livingston, M., Weiss, R., Zhao, H., Haynes, K. A., Morsut, L., Chen, Y. Y., Khalil, A. S., Wong, W. W., Collins, J. J., Rosser, S. J., Polizzi, K., Elowitz, M. B., Fussenegger, M., Hilton, I. B., Leonard, J. N., Bintu, L., Galloway, K. E. and Deans, T. L. (2022), 'The sound of silence: Transgene silencing in mammalian cell engineering', *Cell Systems* **13**(12), 950–973.
URL: *https://www.sciencedirect.com/science/article/pii/S2405471222004641*

24. Castets, V., Dulos, E., Boissonade, J. and De Kepper, P. (1990), 'Experimental evidence of a sustained standing Turing-type nonequilibrium chemical pattern', *Phys. Rev. Lett.* **64**, 2953–2956.
URL: *https://link.aps.org/doi/10.1103/PhysRevLett.64.2953*

25. Chen, Y., Ho, J. M. L., Shis, D. L., Gupta, C., Long, J., Wagner, D. S., Ott, W., Josić, K. and Bennett, M. R. (2018), 'Tuning the dynamic range of bacterial promoters regulated by ligand-inducible transcription factors', *Nature Communications* **9**(1), 64.
URL: *https://doi.org/10.1038/s41467-017-02473-5*

26. Chen, Z., Boyken, S. E., Jia, M., Busch, F., Flores-Solis, D., Bick, M. J., Lu, P., VanAernum, Z. L., Sahasrabuddhe, A., Langan, R. A., Bermeo, S., Brunette, T. J., Mulligan, V. K., Carter, L. P., DiMaio, F., Sgourakis, N. G., Wysocki, V. H. and Baker, D. (2019), 'Programmable design of orthogonal protein heterodimers', *Nature* **565**(7737), 106–111.
URL: *https://doi.org/10.1038/s41586-018-0802-y*

27. Chen, Z., Linton, J. M., Zhu, R. and Elowitz, M. B. (2022), 'A synthetic protein-level neural network in mammalian cells', *bioRxiv* .
URL: *https://www.biorxiv.org/content/early/2022/07/11/2022.07.10.499405*

28. Cherry, K. M. and Qian, L. (2018), 'Scaling up molecular pattern recognition with dna-based winner-take-all neural networks', *Nature* **559**(7714), 370–376.
URL: *https://doi.org/10.1038/s41586-018-0289-6*

29. Cotterell, J., Robert-Moreno, A. and Sharpe, J. (2015), 'A local, self-organizing reaction-diffusion model can explain somite patterning in embryos', *Cell systems* **1**(4), 257–269.

30. Cox, R. S. r., Surette, M. G. and Elowitz, M. B. (2007), 'Programming gene expression with combinatorial promoters.', *Mol Syst Biol* **3**, 145.

31. Crank, J. and Nicolson, P. (1947), 'A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type', *Mathematical Proceedings of the Cambridge Philosophical Society* **43**(1), 50–67.

32. Daniel, R., Rubens, J. R., Sarpeshkar, R. and Lu, T. K. (2013), 'Synthetic analog computation in living cells', *Nature* **497**(7451), 619–623.
URL: *https://doi.org/10.1038/nature12148*

33. Danino, T., Mondragón-Palomino, O., Tsimring, L. and Hasty, J. (2010), 'A synchronized quorum of genetic clocks', *Nature* **463**(7279), 326–330.
URL: *https://pubmed.ncbi.nlm.nih.gov/20090747*

34. de Souza, N. (2018), 'Method of the year 2017: Organoids', *Nature methods* **15**(1), 1–1.

35. Der, B. S., Glassey, E., Bartley, B. A., Enghuus, C., Goodman, D. B., Gordon, D. B., Voigt, C. A. and Gorochowski, T. E. (2017), 'DNAplotlib: Programmable visualization of genetic designs and associated data', *ACS synthetic biology* **6**(7), 1115.

36. Diambra, L., Senthivel, Vivek Raj, ., Bárcena Menéndez, Diego, . and Isalan, M. (2015), 'Cooperativity to increase Turing pattern space for synthetic biology'.

37. DiAndreth, B., Wauford, N., Hu, E., Palacios, S. and Weiss, R. (2022), 'Persist platform provides programmable rna regulation using crispr endornases', *Nature Communications* **13**(1), 2582.
URL: *https://doi.org/10.1038/s41467-022-30172-3*

38. Du, P., Zhao, H., Zhang, H., Wang, R., Huang, J., Tian, Y., Luo, X., Luo, X., Wang, M., Xiang, Y., Qian, L., Chen, Y., Tao, Y. and Lou, C. (2020), 'De novo design of an intercellular signaling toolbox for multi-channel cell–cell communication and biological computation', *Nature Communications* **11**(1), 4226.
URL: *https://doi.org/10.1038/s41467-020-17993-w*

39. Elowitz, M. B. and Leibler, S. (2000), 'A synthetic oscillatory network of transcriptional regulators', *Nature* **403**(6767), 335–338.
URL: *https://doi.org/10.1038/35002125*

40. Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S. and Dehmer, M. (2020), 'An introductory review of deep learning for prediction models with big data', *Frontiers in Artificial Intelligence* **3**.
URL: *https://www.frontiersin.org/article/10.3389/frai.2020.00004*

41. Espah Borujeni, A., Zhang, J., Doosthosseini, H., Nielsen, A. A. K. and Voigt, C. A. (2020), 'Genetic circuit characterization by inferring rna polymerase movement and ribosome usage', *Nature Communications* **11**(1), 5001.
URL: *https://doi.org/10.1038/s41467-020-18630-2*

42. Evans, R., Dal Poggetto, G., Nilsson, M. and Morris, G. A. (2018), 'Improving the interpretation of small molecule diffusion coefficients', *Analytical Chemistry* **90**(6), 3987–3994.
URL: *https://doi.org/10.1021/acs.analchem.7b05032*

43. Ferreira, J. P., Overton, K. W. and Wang, C. L. (2013), 'Tuning gene expression with synthetic upstream open reading frames', *Proceedings of the National Academy of Sciences* **110**(28), 11284–11289.
URL: *https://www.pnas.org/doi/abs/10.1073/pnas.1305590110*

44. Fick, A. (1855), 'Ueber diffusion', *Annalen der Physik* **170**(1), 59–86.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.18551700105*

45. Fink, T., Lonzarić, J., Praznik, A., Plaper, T., Merljak, E., Leben, K., Jerala, N., Lebar, T., Strmšek, Ž., Lapenta, F., Benčina, M. and Jerala, R. (2019), 'Design of fast proteolysis-based signaling and logic circuits in mammalian cells', *Nature Chemical Biology* **15**(2), 115–122.
URL: *https://doi.org/10.1038/s41589-018-0181-6*

46. Forde, J. E. (2005), Delay Differential Equation Models in Mathematical Biology, PhD thesis, University of Michigan.

47. Gaidukov, L., Wroblewska, L., Teague, B., Nelson, T., Zhang, X., Liu, Y., Jagtap, K., Mamo, S., Tseng, W. A., Lowe, A., Das, J., Bandara, K., Baijuraj, S., Summers, N. M., Lu, T. K., Zhang, L. and Weiss, R. (2018), 'A multi-landing pad DNA integration platform for mammalian cell engineering.', *Nucleic Acids Res* **46**(8), 4072–4086.

48. Gam, J. J., DiAndreth, B., Jones, R. D., Huh, J. and Weiss, R. (2019), 'A 'poly-transfection' method for rapid, one-pot characterization and optimization of genetic systems', *Nucleic Acids Research* **47**(18), e106–e106.
URL: *https://doi.org/10.1093/nar/gkz623*

49. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W. and Keutzer, K. (2021), 'A survey of quantization methods for efficient neural network inference'.

50. Gibson, D. G., Young, L., Chuang, R.-Y., Venter, J. C., Hutchison, C. A. and Smith, H. O. (2009), 'Enzymatic assembly of DNA molecules up to several hundred kilobases', *Nature Methods* **6**(5), 343–345.
URL: *https://doi.org/10.1038/nmeth.1318*

51. Gierer, A. and Meinhardt, H. (1972), 'A theory of biological pattern formation', *Kybernetik* **12**(1), 30–39.
URL: *https://doi.org/10.1007/BF00289234*

52. Glock, P., Ramm, B., Heermann, T., Kretschmer, S., Schweizer, J., Mücksch, J., Alagöz, G. and Schwille, P. (2019), 'Stationary patterns in a two-protein reaction-diffusion system', *ACS Synthetic Biology* **8**(1), 148–157. cited By 0.
URL: *https://www.scopus.com/inward/record.uri?eid=2-s2.0-85059756251*

53. Graner, F. m. c. and Glazier, J. A. (1992), 'Simulation of biological cell sorting using a two-dimensional extended Potts model', *Phys. Rev. Lett.* **69**, 2013–2016.
URL: *https://link.aps.org/doi/10.1103/PhysRevLett.69.2013*

54. Green, M. R. M. R. (2012), *Molecular cloning : a laboratory manual. Volume 2*, 4th ed. edn, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, N.Y.

55. Grieneisen, V., Xu, J., Marée, A., Hogeweg, P. and Scheres, B. (2007), 'Auxin transport is sufficient to generate a maximumand gradient guiding root growth'.

56. Guye, P., Ebrahimkhani, M. R., Kipniss, N., Velazquez, J. J., Schoenfeld, E., Kiani, S., Griffith, L. G. and Weiss, R. (2016), 'Genetically engineering self-organization of human pluripotent stem cells into a liver bud-like tissue using Gata6', *Nature Communications* **7**(1), 10243.
URL: *https://doi.org/10.1038/ncomms10243*

57. Hajnal, A., Maass, W., Pudlák, P., Szegedy, M. and Turán, G. (1993), 'Threshold circuits of bounded depth', *Journal of Computer and System Sciences* **46**(2), 129–154.
URL: *https://www.sciencedirect.com/science/article/pii/002200009390001D*

58. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. (2020), 'Array programming with NumPy', *Nature* **585**(7825), 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

59. Hayashi, T. and Carthew, R. W. (2004), 'Surface mechanics mediate pattern formation in the developing retina', *Nature* **431**(7009), 647–652.
URL: *https://doi.org/10.1038/nature02952*

60. Hofer, M. and Lutolf, M. P. (2021), 'Engineering organoids', *Nature reviews. Materials* **6**(5), 402–420.
URL: *https://pubmed.ncbi.nlm.nih.gov/33623712*

61. Hornik, K. (1991), 'Approximation capabilities of multilayer feedforward networks', *Neural Networks* **4**(2), 251–257.
URL: *https://www.sciencedirect.com/science/article/pii/089360809190009T*

62. Horváth, J., Szalai, I. and De Kepper, P. (2009), 'An experimental design method leading to chemical Turing patterns', *Science* **324**(5928), 772–775.
URL: *https://science.sciencemag.org/content/324/5928/772*

63. Isalan, M., Lemerle, C. and Serrano, L. (2005), 'Engineering gene networks to emulate Drosophila embryonic pattern formation'.

64. Jinek, M., Chylinski, K., Fonfara, I., Hauer, M., Doudna, J. A. and Charpentier, E. (2012), 'A programmable dual-rna-guided DNA endonuclease in adaptive bacterial immunity', *Science* **337**(6096), 816–821.
URL: *https://www.science.org/doi/abs/10.1126/science.1225829*

65. Karig, D., Martini, K. M., Lu, T., Delateur, N. A., Goldenfeld, N. and Weiss, R. (2018), 'Stochastic Turing patterns in a synthetic bacterial population', *Proceedings of the National Academy of Sciences of the United States of America* **115**(26), 6572.

66. Kim, J., Koo, B.-K. and Knoblich, J. A. (2020), 'Human organoids: model systems for human biology and medicine', *Nature reviews. Molecular cell biology* **21**(10), 571–584.

67. Kitada, T., DiAndreth, B., Teague, B. and Weiss, R. (2018), 'Programming gene and engineered-cell therapies with synthetic biology', *Science* **359**(6376), eaad1067.
URL: *https://www.science.org/doi/abs/10.1126/science.aad1067*

68. Klika, V., Baker, R. E., Headon, D. and Gaffney, E. A. (2012), 'The influence of receptor-mediated interactions on reaction-diffusion mechanisms of cellular self-organisation', *Bulletin of Mathematical Biology* **74**(4), 935–957.
URL: *https://doi.org/10.1007/s11538-011-9699-4*

69. Klika, V. and Gaffney, E. A. (2017), 'History dependence and the continuum approximation breakdown: the impact of domain growth on Turing's instability', *Proceedings of the Royal Society. A, Mathematical, physical, and engineering sciences* **473**(2199), 1–19.

70. Kogenaru, M. and Tans, S. J. (2014), 'An improved Escherichia coli strain to host gene regulatory networks involving both the AraC and LacI inducible transcription factors', *Journal of Biological Engineering* **8**(1), 2.

URL: *https://doi.org/10.1186/1754-1611-8-2*

71. Kondo, S. and Miura, T. (2010), 'Reaction-diffusion model as a framework for understanding biological pattern formation', *Science (New York, N.Y.)* **329**(5999).

72. Konow, C., Somberg, N. H., Chavez, J., Epstein, I. R. and Dolnik, M. (2019), 'Turing patterns on radially growing domains: experiments and simulations', *Phys. Chem. Chem. Phys.* **21**, 6718–6724.
URL: *http://dx.doi.org/10.1039/C8CP07797E*

73. Kærn, M. and Weiss, R. (2006), Synthetic gene regulatory systems, *in* 'System Modeling in Cellular Biology: From Concepts to Nuts and Bolts', The MIT Press.
URL: *https://doi.org/10.7551/mitpress/9780262195485.003.0013*

74. LaFleur, T. L., Hossain, A. and Salis, H. M. (2022), 'Automated model-predictive design of synthetic promoters to control transcriptional profiles in bacteria', *Nature Communications* **13**(1), 5159.
URL: *https://doi.org/10.1038/s41467-022-32829-5*

75. Lakshmanan, M. and Senthilkumar, D. V. (2010), *Dynamics of Nonlinear Time-Delay Systems*, Springer Series in Synergetics, Springer Berlin Heidelberg, Berlin, Heidelberg.

76. Lindenmayer, A. (1968), 'Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs', *Journal of Theoretical Biology* **18**(3), 300–315.
URL: *https://www.sciencedirect.com/science/article/pii/0022519368900805*

77. Liu, C., Fu, X., Liu, L., Ren, X., Chau, C. K., Li, S., Xiang, L., Zeng, H., Chen, G., Tang, L.-H., Lenz, P., Cui, X., Huang, W., Hwa, T. and Huang, J.-D. (2011), 'Sequential establishment of stripe patterns in an expanding cell population', *Science* **334**(6053), 238–241.
URL: *https://science.sciencemag.org/content/334/6053/238*

78. Lorenz, R., Bernhart, S. H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F. and Hofacker, I. L. (2011), 'ViennaRNA package 2.0', *Algorithms for Molecular Biology* **6**(1), 26.
URL: *https://doi.org/10.1186/1748-7188-6-26*

79. Madzvamuse, A., Gaffney, E. A. and Maini, P. K. (2010), 'Stability analysis of non-autonomous reaction-diffusion systems: the effects of growing domains', *Journal of mathematical biology* **61**(1), 133–164.

80. Maini, P. K., Woolley, T. E., Baker, R. E., Gaffney, E. A. and Lee, S. S. (2012*a*), 'Turing's model for biological pattern formation and the robustness problem', *Interface focus* **2**(4), 487–496.

81. Maini, P. K., Woolley, T. E., Baker, R. E., Gaffney, E. A. and Lee, S. S. (2012*b*), 'Turing's model for biological pattern formation and the robustness problem', *Journal of the Royal Society Interface Focus* **2**(4), 487–496.

82. Manna, K., Volpert, V. and Banerjee, M. (2021), 'Pattern formation in a three-species cyclic competition model', *Bulletin of Mathematical Biology* **83**(5), 52.
URL: *https://doi.org/10.1007/s11538-021-00886-4*

83. McAdams, H. H. and Arkin, A. (1999), 'It's a noisy business! genetic regulation at the nanomolar scale', *Trends in Genetics* **15**(2), 65–69.
URL: *https://doi.org/10.1016/S0168-9525(98)01659-X*

84. McCulloch, W. S. and Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics* **5**(4), 115–133.
URL: *https://doi.org/10.1007/BF02478259*

85. McNerney, M. P., Doiron, K. E., Ng, T. L., Chang, T. Z. and Silver, P. A. (2021), 'Theranostic cells: emerging clinical applications of synthetic biology', *Nature Reviews Genetics* **22**(11), 730–746.
URL: *https://doi.org/10.1038/s41576-021-00383-3*

86. Mead, C. (1990), 'Neuromorphic electronic systems', *Proceedings of the IEEE* **78**(10), 1629–1636.

87. Meng, F. and Ellis, T. (2020), 'The second decade of synthetic biology: 2010–2020', *Nature Communications* **11**(1), 5174.
URL: *https://doi.org/10.1038/s41467-020-19092-2*

88. Meyer, A. J., Segall-Shapiro, T. H., Glassey, E., Zhang, J. and Voigt, C. A. (2019), 'Escherichia coli "Marionette" strains with 12 highly optimized small-molecule sensors', *Nature chemical biology* **15**(2), 196.

89. Milo, R., Jorgensen, P., Moran, U., Weber, G. and Springer, M. (2010), 'Bionumbers–the database of key numbers in molecular and cell biology.', *Nucleic Acids Res* **38**(Database issue), D750–3.

90. Minsky, M., Papert, S. and Bottou, L. (2017), *Perceptrons : an introduction to computational geometry*, [2017 edition]. edn, The MIT Press, Cambridge, Massachusetts.

91. Mordvintsev, A., Randazzo, E., Niklasson, E. and Levin, M. (2020), 'Growing neural cellular automata', *Distill* . https://distill.pub/2020/growing-ca.

92. Murphy, K. P. (2013), *Machine learning : a probabilistic perspective*, MIT Press, Cambridge, Mass.

[u.a.].
URL: *https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr*$_{12}$*?ie* = $UTF8qid = 1336857747sr = 8 - 2$

93. Murphy, K. P. (2022), *Probabilistic Machine Learning: An introduction*, MIT Press.
URL: *probml.ai*

94. Murray, J. D. (2002*a*), *Mathematical Biology: 2. Spatial Models and Biomedical Applications*, third edition edn, Springer.

95. Murray, J. D. (2002*b*), *Mathematical Biology: I. An Introduction*, third edition edn, Springer.

96. Nakamasu, A., Takahashi, G., Kanbe, A. and Kondo, S. (2009), 'Interactions between zebrafish pigment cells responsible for the generation of Turing patterns', *Proceedings of the National Academy of Sciences of the United States of America* **106**(21), 8429.

97. Nelson, J. W., Randolph, P. B., Shen, S. P., Everette, K. A., Chen, P. J., Anzalone, A. V., An, M., Newby, G. A., Chen, J. C., Hsu, A. and Liu, D. R. (2022), 'Engineered pegRNAs improve prime editing efficiency.', *Nature Biotechnology* **40**(3), 402–410.

98. Nielsen, A. A. K., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D. and Voigt, C. A. (2016), 'Genetic circuit design automation', *Science* **352**(6281), aac7341.
URL: *https://www.science.org/doi/abs/10.1126/science.aac7341*

99. Okumura, S., Gines, G., Lobato-Dauzier, N., Baccouche, A., Deteix, R., Fujii, T., Rondelez, Y. and Genot, A. J. (2022), 'Nonlinear decision-making with enzymatic neural networks', *Nature* **610**(7932), 496–501.
URL: *https://doi.org/10.1038/s41586-022-05218-7*

100. Oliver Huidobro, M., Tica, J., Wachter, G. K. A. and Isalan, M. (2021), 'Synthetic spatial patterning in bacteria: advances based on novel diffusible signals', *Microbial biotechnology* .

101. Ouyang, Q. and Swinney, H. L. (1991), 'Transition from a uniform state to hexagonal and striped Turing patterns', *Nature* **352**(6336), 610–612.
URL: *https://doi.org/10.1038/352610a0*

102. Pandi, A., Koch, M., Voyvodic, P. L., Soudier, P., Bonnet, J., Kushwaha, M. and Faulon, J.-L. (2019), 'Metabolic perceptrons for neural computing in biological systems', *Nature Communications* **10**(1), 3880.
URL: *https://doi.org/10.1038/s41467-019-11889-0*

103. Plaza, R., Sánchez-Garduño, F., Padilla, P., Barrio, R. and Maini, P. (2004), 'The effect of growth and curvature on pattern formation', *Journal of dynamics and differential equations* **16**(4), 1093–1121.

104. Potts, R. B. (1952), 'Some generalized order-disorder transformations', *Mathematical Proceedings of the Cambridge Philosophical Society* **48**(1), 106–109.

105. Potvin-Trottier, L., Lord, N. D., Vinnicombe, G. and Paulsson, J. (2016), 'Synchronous long-term oscillations in a synthetic gene circuit', *Nature* **538**(7626), 514–517.
URL: *https://pubmed.ncbi.nlm.nih.gov/27732583*

106. Prusinkiewicz, P. and Lindenmayer, A. (1990), *The Algorithmic Beauty of Plants*, 1 edn, Springer, New York, NY.

107. Purvis, J. E. and Lahav, G. (2013), 'Encoding and decoding cellular information through signaling dynamics', *Cell* **152**(5), 945–956.
URL: *https://doi.org/10.1016/j.cell.2013.02.005*

108. Puzyrev, D., Yanchuk, S., Vladimirov, A. and Gurevich, S. (2013), 'Stability of plane wave solutions in complex Ginzburg-Landau equation with delayed feedback', *SIAM Journal on Applied Dynamical Systems [electronic only]* **13**.

109. Qian, L., Winfree, E. and Bruck, J. (2011), 'Neural network computation with DNA strand displacement cascades', *Nature* **475**(7356), 368–372.
URL: *https://doi.org/10.1038/nature10262*

110. Racovita, A., Prakash, S., Varela, C., Walsh, M., Galizi, R., Isalan, M. and Jaramillo, A. (2022), 'Engineered gene circuits capable of reinforcement learning allow bacteria to master gameplaying', *bioRxiv* .
URL: *https://www.biorxiv.org/content/early/2022/06/21/2022.04.22.489191*

111. Rani, S. A., Pitts, B. and Stewart, P. S. (2005), 'Rapid diffusion of fluorescent tracers into Staphylococcus epidermidis biofilms visualized by time lapse microscopy', *Antimicrobial Agents and Chemotherapy* **49**(2), 728–732.
URL: *https://aac.asm.org/content/49/2/728*

112. Raspopovic, J., Marcon, L., Russo, L. and Sharpe, J. (2014), 'Modeling digits. digit patterning is

controlled by a Bmp-Sox9-Wnt Turing network modulated by morphogen gradients', *Science (New York, N.Y.)* **345**(6196), 566.

113. Riglar, D. T., Richmond, D. L., Potvin-Trottier, L., Verdegaal, A. A., Naydich, A. D., Bakshi, S., Leoncini, E., Lyon, L. G., Paulsson, J. and Silver, P. A. (2019), 'Bacterial variability in the mammalian gut captured by a single-cell synthetic oscillator', *Nature Communications* **10**(1), 4665.
URL: *https://doi.org/10.1038/s41467-019-12638-z*

114. Rizik, L., Danial, L., Habib, M., Weiss, R. and Daniel, R. (2022), 'Synthetic neuromorphic computing in living cells', *Nature Communications* **13**(1), 5602.
URL: *https://doi.org/10.1038/s41467-022-33288-8*

115. Rodnina, M. V. (2016), 'The ribosome in action: Tuning of translational efficiency and protein folding', *Protein science : a publication of the Protein Society* **25**(8), 1390–1406.
URL: *https://pubmed.ncbi.nlm.nih.gov/27198711*

116. Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychol Rev* **65**(6), 386–408.

117. Roth, W., Schindler, G., Fröning, H. and Pernkopf, F. (2020), Training discrete-valued neural networks with sign activations using weight distributions, *in* U. Brefeld, E. Fromont, A. Hotho, A. Knobbe, M. Maathuis and C. Robardet, eds, 'Machine Learning and Knowledge Discovery in Databases', Springer International Publishing, Cham, pp. 382–398.

118. Salis, H. M. (2011), Chapter two - the ribosome binding site calculator, *in* C. Voigt, ed., 'Synthetic Biology, Part B', Vol. 498 of *Methods in Enzymology*, Academic Press, pp. 19–42.
URL: *https://www.sciencedirect.com/science/article/pii/B9780123851208000024*

119. Santos-Moreno, J. and Schaerli, Y. (2019), 'Using synthetic biology to engineer spatial patterns', *Advanced biosystems* **3**(4), 1800280–n/a.

120. Sauro, H. M. and Kim, K. (2013), 'It's an analog world', *Nature* **497**(7451), 572–573.
URL: *https://doi.org/10.1038/nature12246*

121. Schaerli, Y., Munteanu, A., Gili, M., Cotterell, J., Sharpe, J. and Isalan, M. (2014), 'A unified design space of synthetic stripe-forming networks', *Nature communications* **5**(1), 4905–4905.

122. Schnakenberg, J. (1979), 'Simple chemical reaction systems with limit cycle behaviour', *Journal of Theoretical Biology* **81**(3), 389–400.
URL: *https://www.sciencedirect.com/science/article/pii/0022519379900420*

123. Scholefield, J. and Harrison, P. T. (2021), 'Prime editing –an update on the field', *Gene Therapy* **28**(7), 396–401.
URL: *https://doi.org/10.1038/s41434-021-00263-9*

124. Scholes, N. S. and Isalan, M. (2017), 'A three-step framework for programming pattern formation', *Current Opinion in Chemical Biology* **40**, 1 – 7. Synthetic Biology * Synthetic Biomolecules.
URL: *http://www.sciencedirect.com/science/article/pii/S1367593117300108*

125. Scholes, N. S., Schnoerr, D., Isalan, M. and Stumpf, M. P. H. (2019), 'A comprehensive network atlas reveals that Turing patterns are common but not robust'.

126. Schrödinger, E. (1948), *What Is Life? The Physical Aspect of the Living Cell*, Cambridge University Press.

127. Schuman, C. D., Kulkarni, S. R., Parsa, M., Mitchell, J. P., Date, P. and Kay, B. (2022), 'Opportunities for neuromorphic computing algorithms and applications', *Nature Computational Science* **2**(1), 10–19.
URL: *https://doi.org/10.1038/s43588-021-00184-y*

128. Sekine, R., Shibata, T. and Ebisuya, M. (2018), 'Synthetic mammalian pattern formation driven by differential diffusivity of Nodal and Lefty', *Nature Communications* **9**(1), 54–56.
URL: *https://doi.org/10.1038/s41467-018-07847-x*

129. Sen, S., Ghosh, P., Riaz, S. S. and Ray, D. S. (2009), 'Time-delay-induced instabilities in reaction-diffusion systems.', *Phys Rev E Stat Nonlin Soft Matter Phys* **80**(4 Pt 2), 046212.

130. Shakiba, N., Jones, R. D., Weiss, R. and Del Vecchio, D. (2021), 'Context-aware synthetic biology by controller design: Engineering the mammalian cell', *Cell Systems* **12**(6), 561–592.
URL: *https://www.sciencedirect.com/science/article/pii/S2405471221001976*

131. Srivastava, S., Divekar, A. V., Anilkumar, C., Naik, I., Kulkarni, V. and Pattabiraman, V. (2021), 'Comparative analysis of deep learning image detection algorithms', *Journal of Big Data* **8**(1), 66.
URL: *https://doi.org/10.1186/s40537-021-00434-w*

132. Storch, M., Casini, A., Mackrow, B., Fleming, T., Trewhitt, H., Ellis, T. and Baldwin, G. S. (2015), 'BASIC: A new biopart assembly standard for idempotent cloning provides accurate, single-tier DNA assembly for synthetic biology', *ACS synthetic biology* **4**(7), 781–787.

133. Strogatz, S. H. S. H. (2000), *Nonlinear dynamics and chaos : with applications to physics, biology,*

chemistry, and engineering, Studies in nonlinearity, Westview Press, Cambridge, Mass.

134. Tamsir, A., Tabor, J. J. and Voigt, C. A. (2011), 'Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'', *Nature* **469**(7329), 212–215.
URL: *https://doi.org/10.1038/nature09565*

135. Teague, B. P., Guye, P. and Weiss, R. (2016), 'Synthetic morphogenesis', *Cold Spring Harbor perspectives in biology* **8**(9), a023929–.

136. Teague, B. and Weiss, R. (2017–2022), 'Cytoflow: Python tools for quantitative, reproducible flow cytometry analysis', `https://cytoflow.github.io`.

137. Thompson, D. W. (1942), *On growth and form Volume II*, new ed. edn, Cambridge University Press, Cambridge.

138. Tian, C. and Zhang, L. (2013), 'Delay-driven irregular spatiotemporal patterns in a plankton system', *Phys. Rev. E* **88**, 012713.
URL: *https://link.aps.org/doi/10.1103/PhysRevE.88.012713*

139. Tordoff, J., Krajnc, M., Walczak, N., Lima, M., Beal, J., Shvartsman, S. and Weiss, R. (2021), 'Incomplete cell sorting creates engineerable structures with long-term stability', *Cell Reports Physical Science* **2**, 100305.

140. Turing, A. M. (1952), 'The chemical basis of morphogenesis', *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* **237**(641), 37–72.
URL: *http://www.jstor.org/stable/92463*

141. Ulmann, B. (2022), *Analog Computing*, 2 edn, De Gruyter Oldenbourg, https://doi.org/10.1515/9783110787740.

142. van Gestel, J., Bareia, T., Tenennbaum, B., Dal Co, A., Guler, P., Aframian, N., Puyesky, S., Grinberg, I., D'Souza, G. G., Erez, Z., Ackermann, M. and Eldar, A. (2021), 'Short-range quorum sensing controls horizontal gene transfer at micron scale in bacterial communities', *Nature Communications* **12**(1), 2324.
URL: *https://doi.org/10.1038/s41467-021-22649-4*

143. Van Gorder, R. A., Klika, V. and Krause, A. L. (2021), 'Turing conditions for pattern forming systems on evolving manifolds', *Journal of mathematical biology* **82**(1-2), 4–4.

144. Vassallo, L., Hansmann, D. and Braunstein, L. A. (2019), 'On the growth of non-motile bacteria colonies: an agent-based model for pattern formation', *The European physical journal. B, Condensed matter physics* **92**(9), 1–8.

145. Vohradsky, J. (2001), 'Neural network model of gene expression', *The FASEB Journal* **15**(3), 846–854.
URL: *https://faseb.onlinelibrary.wiley.com/doi/abs/10.1096/fj.00-0361com*

146. Wang, S., Garcia-Ojalvo, J. and Elowitz, M. B. (2022), 'Periodic spatial patterning with a single morphogen', *bioRxiv* .
URL: *https://www.biorxiv.org/content/early/2022/03/22/2022.03.21.484932*

147. Wauford, N., Jones, R., Van De Mark, C. and Weiss, R. (2023), 'Rapid development of cell state identification circuits with poly-transfection.', *J Vis Exp* (192).

148. Weber, E., Engler, C., Gruetzner, R., Werner, S. and Marillonnet, S. (2011), 'A modular cloning system for standardized assembly of multigene constructs', *PloS one* **6**(2), e16765–e16765.

149. Wegener, I. (1991), 'The complexity of the parity function in unbounded fan-in, unbounded depth circuits', *Theoretical Computer Science* **85**(1), 155–170.
URL: *https://www.sciencedirect.com/science/article/pii/0304397591900524*

150. Wolpert, L. (1969), 'Positional information and the spatial pattern of cellular differentiation', *Journal of Theoretical Biology* **25**(1), 1–47.
URL: *https://www.sciencedirect.com/science/article/pii/S0022519369800160*

151. Wolpert, L. (2016), 'Positional information and pattern formation', *Current topics in developmental biology* **117**, 597–608.

152. Xie, Z., Wroblewska, L., Prochazka, L., Weiss, R. and Benenson, Y. (2011), 'Multi-input RNAi-based logic circuit for identification of specific cancer cells', *Science* **333**(6047), 1307–1311.
URL: *https://www.science.org/doi/abs/10.1126/science.1205527*

153. Yi, F., Gaffney, E. A. and Seirin-Lee, S. (2017), 'The bifurcation analysis of Turing pattern formation induced by delay and diffusion in the schnakenberg system', *Discrete & Continuous Dynamical Systems - B* **22**(1531-3492-2017-2-647), 647.
URL: *http://aimsciences.org//article/id/18d1a7f3-4208-4c5f-9080-95673ac6bb6a*

154. Zhang, S., Wan, Z. and Kamm, R. D. (2021), 'Vascularized organoids on a chip: strategies for engineering organoids with functional vasculature', *Lab Chip* **21**, 473–488.
URL: *http://dx.doi.org/10.1039/D0LC01186J*

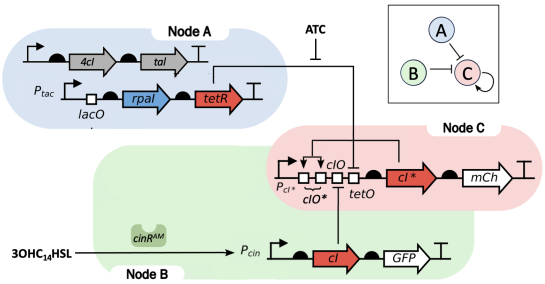
155. Zhang, T. and Zang, H. (2014), 'Delay-induced Turing instability in reaction-diffusion equations.', *Phys Rev E Stat Nonlin Soft Matter Phys* **90**(5-1), 052908.

156. Zhao, N., Song, Y., Xie, X., Zhu, Z., Duan, C., Nong, C., Wang, H. and Bao, R. (2023), 'Synthetic biology-inspired cell engineering in diagnosis, treatment, and drug development', *Signal Transduction and Targeted Therapy* **8**(1), 112.
URL: *https://doi.org/10.1038/s41392-023-01375-x*

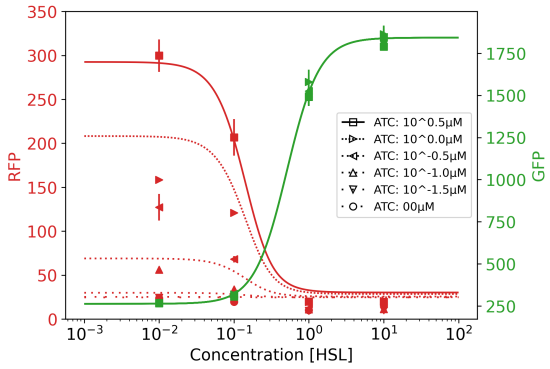
157. Zuniga, A. and Zeller, R. (2014), 'In Turing's hands: the making of digits', *Science* **345**(6196), 516–517.
URL: *https://www.science.org/doi/abs/10.1126/science.1257501*

Appendix A Additional Liquid Culture Data for Circuit 3954 controls

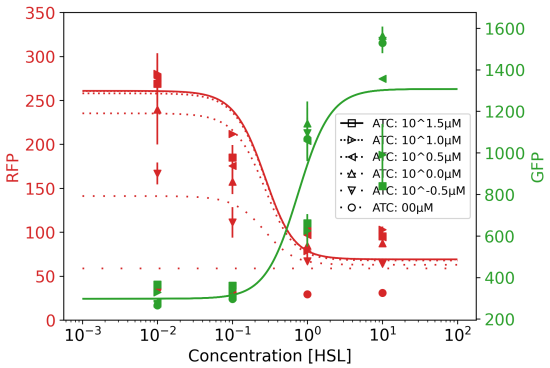
Additional datasets for the liquid culture data described in Section 2.2. The first control is shown in Figure A.72, the second in Figure A.73. The parameters to generate the curves are shown in Table A.15 and Table A.16 respectively.



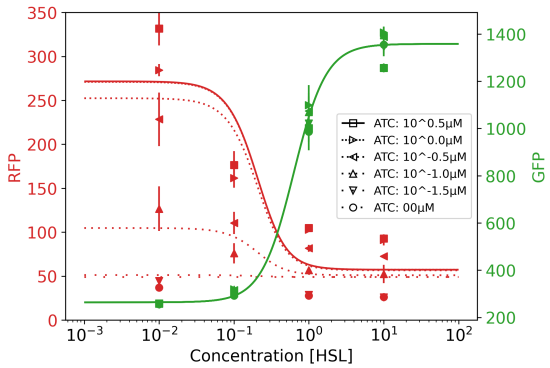
(a) Circuit Schematic



(b) E20201126A



(c) E20201123B

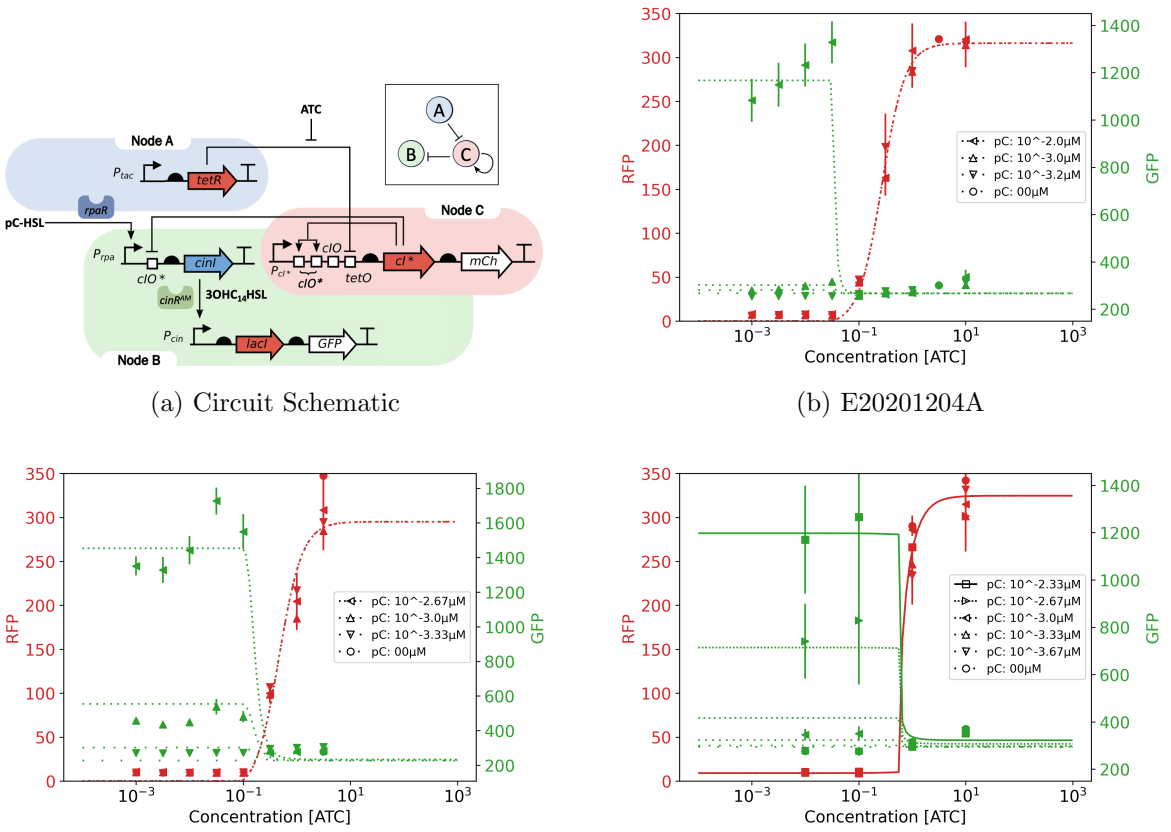


(d) E20201124A

Figure A.72: Additional results for the first control presented in Section 2.2. E20201126A (Figure A.72b) is the same as Figure 13b. The parameters to the corresponding models are shown in Table A.15.

Unique ID	$Vmax_B$	$Vmax_C$	kA_B	kA_{BC}	kA_C	b_B	b_C	kA_{CC}	n_B	n_{BC}	n_C	n_{CC}	MSE
E20201126A	1.58E+03	5.97E+06	5.09E-01	1.81E+00	7.34E-01	2.63E+02	2.47E+01	1.84E-31	2	2	2	1	4.76E+02
E20201123B	1.01E+03	7.05E+02	7.55E-01	4.61E+02	7.91E-02	2.99E+02	6.69E+01	3.23E+02	2	2	2	2	1.32E+04
E20201124A	1.09E+03	1.63E+07	6.27E-01	1.02E+00	1.99E-01	2.64E+02	4.91E+01	1.20E-03	2	2	2	4	1.16E+03

Table A.15: Parameters for the best fit displayed in Figure A.73. MSE is the mean squared error for the fit to the data. Parameters are fit to populate the model described in Equation 12.



(a) Circuit Schematic

(b) E20201204A

(c) E20201210A

(d) E20201203A

Figure A.73: Additional results for the first control presented in Section 2.2. E20201204A is the same as Figure 14b. The parameters to the corresponding models are shown in Table A.16.

Unique ID	$Vmax_C$	kA_C	b_C	kA_{CC}	$Vmax_B$	kA_B	b_B	kA_{CB}	n_B	n_C	n_{CB}	n_{CC}	MSE
E20201204A	3.20E+02	2.73E-01	1.13E-07	4.24E+00	1.20E+03	5.77E-03	2.67E+02	1.15E+00	2	2	2	1	1.17E+03
E20201210A	3.08E+02	4.67E-01	2.74E-10	1.33E+01	5.30E+03	3.90E-03	2.27E+02	2.11E+01	2	2	2	1	3.57E+03
E20201203A	4.09E+02	3.34E-01	9.03E+00	1.77E+02	1.33E+03	3.09E-03	2.94E+02	5.80E+01	2	2	2	2	1.34E+03

Table A.16: Parameters for the best fit displayed in Figure A.73. MSE is the mean squared error for the fit to the data. Parameters are fit to populate the model described in Equation 13.

Appendix B Table of Abbreviations

A complete table of all abbreviations used in this thesis can be found in B.17.

Abbreviation	Definition
(e)pegRNA	(engineered) prime editing guide ribonucleic acid
$3OHC_{14}HSL$	3-hydroxytetradecanoyl-homoserine lactone
AiiA	N-acyl homoserine lactonase
ATC	Anhydrotetracycline
cI	lambda phage cI dual-action transcription factor
*cI**	cI5G6G,P transcription factor variant
cinI	Acyl-homoserine-lactone synthase
DNA	Deoxyribonucleic acid
E. coli	*Escherichia Coli*
ERN	Endoribonuclease
HEK	Human Embryonic Kidney
HeLa	Henrietta Lachs
HSL	Homoserine lactone
IPTG	Isopropyl ß-D-1-thiogalactopyranoside
lacI	Lactose Repressor
mRNA	messenger ribonucleic acid
NMR	Nuclear Magnetic Resonance
ODE	Ordinary Differential Equation
PAM	protospacer adjacent motif
pC-HSL	N-(p-coumaroyl)-L-homoserine lactone
PDE	Partial Differential Equation
PE	Prime editing
PERSIST	Programmable Endonucleolytic Scission-Induced Stability Tuning
RBS	Ribosome Binding Site
RD	Reaction-diffusion
RNA	Ribonucleic acid
rpaI	pC-HSL synthase
SBOL	Synthetic Biology Open Language
tetR	Tetracycline Repressor
uORF	Upstream open reading frame

Table B.17: Table of abbreviations used throughout this thesis.