A π-calculus Model of Supercoiling DNA Circuits

Penn Faulkner Rainford[1], Aalap Mogre[2], Victor Velasco-Berrelleza[3,4], Charles J. Dorman[2], Sarah Harris[3,4], Carsten Kröger[2] and Susan Stepney[1]

[1]Department of Computer Science, University of York, York, UK
[2]Department of Microbiology, School of Genetics and Microbiology, Moyne Institute of Preventive Medicine, Trinity College Dublin, Ireland.
[3]School of Physics and Astronomy, University of Leeds, Leeds, UK
[4]Astbury Centre for Structural and Molecular Biology, University of Leeds, Leeds, UK
penn.rainford@york.ac.uk

Abstract

Synthetic biology is one facet of Artificial Life which designs novel biological components, e.g. DNA, RNA, membranes, to produce new behaviours. Here, we are interested in DNA "circuits": DNA engineered to have particular computational properties. During gene transcription, the DNA double-helix undergoes *supercoiling* changes, which affects transcription of nearby genes. There is limited mathematical, as opposed to physical, modelling of DNA circuits, and supercoiling is not considered. In many current synthetic circuits, supercoiling has to be carefully removed, particularly in *in vivo* systems, to prevent unmodelled side effects. However, supercoiling is an intrinsic property of DNA that impacts gene expression, and could be exploited if included in models. Here, we present a new π-calculus formalism for modelling DNA circuits with supercoiling, and demonstrate its use on a simple genetic circuit. The state transition diagrams normally associated with π-calculus are not accessible when the number of states becomes large. We present a new circular visualisation of the π-calculus circuit components that is more intuitive and readable for biologists familiar with the circular visualisations of plasmids.

Introduction

Synthetic biology is a growing engineering discipline that explores 'life as it could be' in two subfields: using engineered molecules to reproduce emergent behaviours of natural biology, and seeking interchangeable biological parts to assemble into systems with engineered behaviours.

Synthetic biology exploits modern technologies to engineer biological systems including DNA, RNA and membranes (Cardelli, 2005). These entities have been manipulated to build biological instances of binary logic gates (Tamsir et al., 2011) and other computational components such as oscillators (Kim and Winfree, 2011), switches (Lou et al., 2010; Kim et al., 2006) and image edge detectors (Tabor et al., 2009). These are often implemented in *plasmids*, small bacterial replicons where the DNA double helix sequence forms a closed circle.

The DNA sequence is read to generate an RNA molecule, a process called transcription. To be *transcribed*, the DNA double helix has to be *melted*, separating the two strands so information can be read from one strand by the RNA polymerase complex. During transcription DNA undergoes twisting changes, called *supercoiling*, which tightens or loosens the DNA helix, thereby making melting harder or easier (Liu and Wang, 1987). Hence supercoiling affects transcription rates of adjacent genes, changing its behaviour.

Most current formalisms for DNA circuits do not consider supercoiling. In many current synthetic circuits, supercoiling has been carefully removed, particularly from *in vivo* systems, to prevent unmodelled side effects. However, supercoiling is an intrinsic property of DNA that impacts gene expression, and could be exploited if included in models and circuit designs. Here, we present a new π-calculus formalism for modelling DNA circuits that includes supercoiling as a basic component. We demonstrate its use on a simple plasmid based circuit.

The visualisations normally associated with π-calculus, such as state transition diagrams, are not very accessible to biologists who want to design plasmid circuits. Even experts struggle with state transition diagrams when the number of states becomes large. We present a new visualisation of our π-calculus circuit descriptions that is more intuitive and readable for those familiar with the circular component descriptions used in biology. This visualisation uses a circular template to provide information on genetic components, supercoiling regions, controls, and environmental inputs and outputs. It includes the more complex structure of cross-linking, which prevents the propagation of supercoiling by connecting two regions of the DNA to form a figure of eight shape. We refer to this structure as a *bridge*, with the bridge needing to be "open" for supercoiling to propagate past it.

Background and related work

DNA Supercoiling. DNA includes genes that code for proteins through a two stage process: genes are *transcribed* to mRNA by RNA polymerase; mRNA is *translated* to proteins by ribosomes. Not all genes are expressed all the time. The promoter controls expression of the gene. Promoters may require the presence or absence of transcription factors, themselves proteins, in order to operate and allow transcrip-

tion. Sometimes, all they provide is a binding site for the RNA polymerase to initiate transcription.

DNA is a double helix of two polynucleotide strands that coil around each other. For transcription to occur RNA polymerase needs to access the helix and bind to one of the strands. The RNA polymerase transcribes one strand, requiring the two DNA strands to be separated (*melted*) for transcription to be initiated. Making promoter melting easier promotes transcription.

Supercoiling describes whether the DNA strands coil less tightly ('negative supercoiling') or more tightly ('positive supercoiling') than relaxed DNA. Negative supercoiling, unwinding the DNA helix, makes melting easier, and promotes transcription. Supercoiling is interesting as it moves along the DNA much faster than transcription. Supercoiling is generated during transcription: as the RNA polymerase moves along the DNA it creates positive supercoiling ahead of it, and negative supercoiling behind (Liu and Wang, 1987). Supercoiling can diffuse along the DNA and melt the promoters of other genes in the DNA (Kouzine et al., 2008; El Houdaigui et al., 2019), known as Supercoiling Induced Duplex Destabilisation (SIDD) (Bi and Benham, 2004; Wang et al., 2006). Positive supercoiling however can "jam" the machinery of the DNA, if it fails to diffuse, by tightening the coils so much it prevents any transcription. Propagation of supercoiling IS restricted by barriers for eg. DNA bridge formation by DNA binding proteins. Replication origins(Lodge et al., 1989) or transcription of transmembrane protein genes are also thought to act as barriers to diffusion of supercoiling. Supercoiling is prevented from building up in systems by relaxing negative supercoils with topoisomerase I and positive supercoils with DNA gyrase. We propose exploiting supercoiling for computation by using it as a fast signalling system, controlling the expression of different genes based on controlling supercoiling in different regions of the DNA.

DNA circuits. Transcription of DNA responds to inputs like transcription factors, temperature and energy; outputs are mRNA and proteins. These input/output interactions can be used to model gene regulation computationally. There are several different models focused on particular processes. For example, there is the process of single strands of DNA binding with RNA. The strands displace each other based on forming stronger bonds; this can be used to model sustained oscillations (Lakin et al., 2012) and feedback control circuits (Yordanov et al., 2014). These systems are engineered *in vitro*, outside a cell, and have been well defined by the use of π-calculus and a programming language (Phillips and Cardelli, 2009). However, since they are based on single strands they cannot make use of the fast signalling along the DNA provided by supercoiling.

Promoters on a double-stranded DNA helix respond to transcription factors (proteins) as input and produce 'reporter' proteins as output. The promoter-gene combinations can be modelled as logic gates; e.g. a promoter requiring two transcription factors before it activates can be modelled as an AND gate. Expressed proteins that are transcription factors can be modelled as 'wiring' between promoters. This Boolean circuit model is used in synthetic biology to design and synthesise artificial genetic circuits in bacteria such as *Escherichia coli* or *Salmonella enterica*. These have been used to develop NOR gates, which can be built into more complex logic functions such as XOR or EQUALS.

A NOR gate using only one promoter in the DNA can be engineered without the natural supercoiling of the DNA becoming a problem (Tamsir et al., 2011). However when we add multiple promoters to the system, the negative supercoiling generated by transcription can activate genes that would be suppressed under the simple Boolean logic circuit model. There can also be areas of positive supercoiling that "jam" the system. These more complex circuits have to be designed carefully to eliminate these supercoiling effects. This "designing out" of supercoiling has been successful in producing complex computation capable of functions like counting input pulses (Friedland et al., 2009) or edge detection of projected images (Tabor et al., 2009). However, without a model of supercoiling, automated generation of computational circuits of this sort is limited.

A model that accounts for supercoiling, even without using it for fast signalling or as part of the computation and control structures, would improve the ability to generate accurate Boolean logic circuits in DNA. Here, by moving away from Boolean logic descriptions to a signalling based model, we can include both promoter-based regulation and supercoiling in a single model.

Process algebras and π-calculus in biology. Process algebras (Fokkink, 1999) are formalisms designed for modelling concurrent systems, where multiple actions can happen at once. They provide a description language for how component processes interact, signal and synchronise actions. They also describe how components change state over time. π-calculus (Parrow, 2001) is a process algebra that focuses on the use of signalling. In π-calculus we can describe how components update their state based on sending and receiving *synchronised* signals. Communication between components is mutual and synchronous.

π-calculus has been used to describe different systems in biology, including ant behaviour (Sumpter et al., 2001), the biochemistry of molecular systems (Regev et al., 2000; Regev and Shapiro, 2004; Curti et al., 2004; John et al., 2013; Kwiatkowski and Stark, 2008), enzyme cascades (McCaskill and Niemann, 2001), and single strand DNA circuit models (Lakin et al., 2012). This formalism has a visualisation for single stranded DNA circuits. However, that visualisation is not appropriate for supercoiling in DNA circuits. The main component of the visualisation is parallel

and forking strands of DNA, which are not part of the double stranded helix DNA used in our *in vivo* systems.

A Motivating Example

We provide an example of a DNA circuit designed to exploit supercoiling, which we model with π-calculus below. We are also engineering this circuit into a functional plasmid in both *E. coli* and *S. enterica* in the wet-lab.

Our circuit detects the presence of lactose its environment. The pBR322 plasmid based *leu-500* promoter "supercoiling sensor" circuit (fig.1) is designed to produce different fluorescent output in the presence or absence of lactose as sensed by the *lac* repressor (LacI). Different behaviours are produced by the propagation, or not, of transcription-induced supercoiling in the circuit. Two barriers to supercoiling propagation exist: (1) the origin of replication, (2) the anchoring of the plasmid to a membrane by TetA (Lodge et al., 1989). A third barrier to supercoiling diffusion is controlled by two regions defined by *lac* operator sites (*lacO1*), which in the presence of LacI forms a DNA loop insulated from supercoiling effects outside the loop. This prevents supercoiling from moving into the looped-out region thereby preventing $P_{leu\text{-}500}$ activation within this loop.

The supercoiling in this circuit is generated by the constitutive transcription of the *tetA* gene. The *tetA* gene promoter (G_{tetA}) is insensitive to transcription factors or supercoiling, requiring only RNA polymerase to be transcribed. Transcription of *tetA* produces negative supercoiling upstream and positive supercoiling downstream. The bacteria this circuit is made to operate in do not have topoisomerse-I activity (removes negative supercoiling), but have intact DNA-gyrase activity (removes positive supercoiling). Relaxation of the positive but not the negative supercoils produced by *tetA* expression increase the global negative supercoiling of the plasmid thereby activating $P_{leu\text{-}500}$. The first active $P_{leu\text{-}500}$ controls the *mRaspberry* gene, which codes for a red fluorescing protein. The supercoiling continues upstream (anti-clockwise here) before encountering the *lac* operator site. If LacI is bound to lacO1 sites (absence of lactose), then the *lac* operator sites are bridged to form a barrier to diffusion of supercoils and accumulation of negative supercoils is restricted to the region outside the loop thereby preventing mKalama1 expression. If LacI is not bound (presence of lactose), the negative supercoiling activates the second $P_{leu\text{-}500}$ initiating transcription of the *mKalama1* gene which codes for a blue fluorescing protein.

Hence, the controlled propagation of supercoiling is an intrinsic mechanism in the functioning of this circuit.

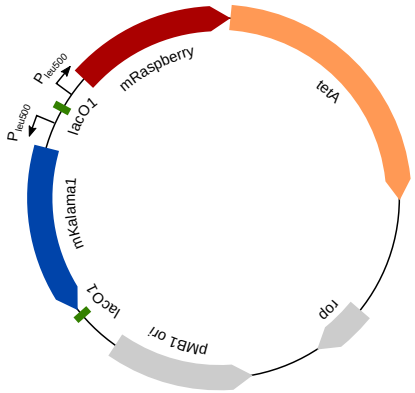

Figure 1: Plasmid map (see Doronina (2019) for plasmid maps) of the *leu-500* promoter supercoiling sensor circuit. Transcription of the *tetA* gene produces upstream (towards the tail of the arrow) negative supercoiling. This reaches the first $P_{leu\text{-}500}$ promoter (black clockwise arrow) that drives the expression of the *mRaspberry* gene. If LacI is not present, supercoiling continues to the second $P_{leu\text{-}500}$ promoter (black anti-clockwise arrow) melting it and activating the expression of the *mKalama1* gene. If LacI *is* present, it binds the two *lacO1* operator sites together, which prevents supercoiling propagating through this barrier. This cuts off the second $P_{leu\text{-}500}$ promoter so it does not experience negative supercoiling, and *mKalama1* is not expressed.

Model

Components

We first define the set of components in our model. We start our components with the different parts of the genome: *tetA*, promoters, *lac* operator sites, promoters of fluorescent proteins, and the origin of replication. We name these as functional components: gene, promoters, bridge, fluorescent and origin respectively. For the modelling in this paper the origin is not required as part of the mathematical model but is included in visualisations.

Supercoiling could be modelled with signalling, which makes the ordering of objects in the genome explicit. However signalling would struggle with topological changes such as that caused by bridging Instead, we allocate components to supercoiling *regions*, with each region having its own supercoiling component. Regions are defined by the components that affect supercoiling, such as genes and bridges, rather than just responding to it.

Finally we add input and output. There is no fixed component that receives all input or provides all output; we model the origins and recipients of signals as further components. In the case of chemical signals outside the circuit, we consider these to come from and go to the "Environment".

Components act together and update concurrently. They send and receive signals and react. At each step, they undergo a state transition, which may include signalling and may change state. Components can take different actions based on their state. For example, a promoter may receive a signal telling it to become negatively supercoiled and change to a state where it increases its output (transcription rate). For objects in the environment, this state change can

track the accumulation of proteins in the environment. For genome components, state is normally linked to the current supercoiling at that component, this is also the case for the supercoiling regions.

π-calculus for supercoiling in plasmids

The requirement for synchronous communication can lead to deadlocking. This happens if a component is not able to act because all its available actions require either sending or receiving a signal but there is no corresponding process available to receive or send the signal. We avoid this issue by providing a non-signalling state change in our models.

In this section, we discuss the basic syntax used in our version of π-calculus. We have four main types of components in our model: Promoters (P), Supercoiling regions (S), Supercoiling barriers (B) and Environment indicators (E). A component has multiple possible states; the current state is indicated by a superscript on the right, e.g. P^*. Here we deal with supercoiling sensitive promoters, a special instance of promoter we call C, which are neutrally coiled, C, or negatively coiled, C^-.

We describe the behaviour of a component with an equation defining how its state changes, either spontaneously or by successfully sending or receiving signals on channels.

The coiling sensitive promoters have two channels, which can be used to send or receive signals. The supercoiling channel s is used for supercoiling signals, to update the component's supercoiling state. Receiving the signal is denoted as $\bar{s}\langle c\rangle$; the coiling value c is used to update the supercoiling state to C^c. The gene channel g is used to signal transcription of the gene with the rate of transcription r. Sending this signal is denoted as $g(r)$; it does not change the supercoiling state of the Promoter. The rate of output r is either low or high, depending on whether the coiling state of the promoter is neutral C or negative C^- respectively. To prevent the model becoming deadlocked, we allow the component to do nothing and stay in the same state. These three options (receive on s, send on r, or do nothing), acting on two initial supercoiling states, are:

$$C := \bar{s}\langle c\rangle.C^c + g(r_{lo}).C + C \quad (1)$$

$$C^- := \bar{s}\langle c\rangle.C^c + g(r_{hi}).C^- + C^- \quad (2)$$

These equations are read as follows: (1) a neutrally coiled promoter C can receive a supercoiling signal $\bar{s}\langle c\rangle$ and transition to C^c, or it can send a gene signal r_{lo} (low because it is neutrally coiled) and stay in state C, or it can do nothing (just stay in state C); (2) a negatively coiled promoter C^- can receive a supercoiling signal $\bar{s}\langle c\rangle$ and transition to C^c, or it can send a gene signal r_{hi} (high because it is negatively coiled) and stay in state C^-, or it can do nothing (just stay in state C^-). The "+" operator denotes the choice between the options; the "." operator indicates that the signals happen and then the state changes.

The π-calculus requires that any signal sent or received by one component must have a corresponding receipt or send from another component. This means there are multiple components running in parallel:

$$circuit := C|E \quad (3)$$

where "|" indicates that the components update their states at the same time.

We use a notational simplification to describe the states of environment E; its state is the amount of the protein in the environment, given as an natural number. (Here we illustrate with a single protein; multiple proteins are introduced later.) Using this notation, we describe all the states, E^i, for $i \in \mathbb{N}$, in a single equation with reference to state i:

$$\begin{aligned} E^i :=& \bar{g}\langle r\rangle.E^{max(i+r-d,0)} \\ &+ p(i).E^{max(i-d,0)} + E^{max(i-d,0)} \end{aligned} \quad (4)$$

Here the environment can do one of three things: it can receive a gene signal $\bar{g}\langle r\rangle$ and increase the amount of that gene in the environment state by r; it can send a signal on the protein channel p signals the amount of protein in the environment; or it can do "nothing". In all cases, the protein in the environment decays at a linear[1] rate d.

If we set the decay rate at zero, $d = 0$, set the promoter output rates at $r_{lo} = 0$ and $r_{hi} = 1$, and run the circuit in eqn(3), starting with C^-, we get:

$$C^-|E^0 \rightarrow C^-|E^1 \rightarrow C^-|E^2 \quad (5)$$

The model so far has a single promoter and a single protein in a single environment. In a full circuit we can have multiple components of these types, and we may need to receive and send multiple signals from different promoters and other components.

For multiple genes, we need a gene channel for each protein generated by the genes; these are named g_X, where X is the gene name. Each protein channel reports the presence of a protein in the environment and is named p_X, where X is the protein name. Multiple barriers and promoters might need to detect a protein. We use a notation simplification to allow a single signal on the protein channel to be read by multiple receivers, as long as there is at least one receiving when it is sent. This notation simplifies our models and makes them easier to read, but the underlying semantics is still separate channels and multiple states.

Additionally, a circuit may contain multiple copies of a gene (not the case in our example circuit). If multiple instances are being transcribed, the environment would need

[1] We use a linear decay rate for simplicity, to match the use of integer values of volume of protein in the environment. When we come to include specific concentrations and timings, we will use a more biologically realistic decay rate. Here, the "+" and "−" in the state calculation are the usual arithmetic operators. The choice of linear decay necessitates the use the max function to ensure that levels do not fall below zero.

to receive both those protein signals. In classic π-calculus this requires two channels with different names being used in parallel with three different options for the state change, two for receiving on just one or other of the channels and the third for receiving on both. We use a further notation simplification to allow a single channel to be used, where all signals sent on the channel at the same time can be received at once, as long as at least one signal is sent.

Promoters

There are two types of promoters used in our example circuits, the *tetA* gene (and promoter) and the supercoiling sensitive promoter controlling the fluorescent protein genes. We are not interested in *tetA* as an input/output of this particular circuit, so it is not tracked and it is always on. In the process of transcription of *tetA* an anchor to a membrane is formed, which causes the transmission of large amount of positive and negative supercoiling to the rest of the circuit. We are not interested in the positive supercoiling, which occurs in a region where it does not affect the rest of our components and is removed by DNA gyrase. We define the *tetA* gene promoter, G_{tetA}, as having no inputs and a large supercoiling negative signal on a coiling channel, $c(-)$. This is an input channel for the region *causing* coiling, as opposed to the supercoiling channel discussed earlier that reports the current state of local supercoiling. For systems with multiple supercoiling regions we match the region's subscript with the channels' subscripts, so supercoiling region S_1 will have a coiling channel c_1 and a supercoiling channel s_1. Since the promoter does not turn on or off, G_{tetA} has only one state; it does have multiple options for state transitions in case it cannot send the supercoiling signal.

$$G_{tetA} := c_1(-).G_{tetA} + G_{tetA} \tag{6}$$

The second type of promoter is a variant of the supercoiling sensitive promoter C. It is the same promoter distinguished due to the fluorescent gene it is promoting. We call it C_{colour}, with the colour of light the fluorescent proteins emit. This makes the computational model more readable as protein names are not always indicative of the fluorescence colour. We have two C fluorescent gene promoters in our example: C_{red} and C_{blue}, with $r_{lo} = 0$, $r_{hi} = 1$.

$$C_{red} := \bar{s_1}\langle c\rangle.C^c_{red} + C_{red} \tag{7}$$

$$C^-_{red} := \bar{s}\langle c\rangle.C^c_{red} + g(1).C^-_{red} + C^-_{red} \tag{8}$$

$$\tag{9}$$

Supercoiling and Supercoiling Regions

These promoters all have supercoiling as inputs/outputs so we need a component to send/receive signals and track the supercoiling in different parts of the DNA. To do so, we define supercoiling regions as a type of component. Each such component deals with a region of the DNA delimited by supercoiling barriers. Not all barriers are always active, so sometimes supercoiling can transmit between regions.

In our example circuit, we have G_{tetA} anchored to a membrane and so acts as a barrier to supercoiling. This defines S_1, the supercoiling region upstream of it. The other barriers in our example are the bridges and the origin of replication. Since we are not interested in the area of DNA downstream of the origin of replication we do not include it. This leaves us with S_1 as the component between G_{tetA} and the first bridge point, and the supercoiling region between the two bridge points, S_2.

A supercoiling region has two possible states: neutrally coiled (relaxed), S_i, and negatively coiled, S_i^-. A region S_i receives a coiling signal on the coiling channel, c_i. Supercoiling regions can receive multiple coiling signals on this channel. It also sends the information on the current state of supercoiling to all components within the region which are sensitive to it on the supercoiling channel, s_i.

$$S^u_i := \bar{c_i}\langle v\rangle.S^v_i + s_i(u).S^u_i + S^u_i \tag{10}$$

Here, we describe the behaviour for all states of S_i where u is the current state and v is the updated state, both drawn from the set $\{+, 0, -\}$ to indicate positive, neutral and negative coiling respectively (neutral coiling is sent as a signal, rather than a lack of signal, as the coiling is relaxed by a separate signal). So the supercoiling region is always either listening for a coil signal to update the supercoiling in the region, or trying to signal its supercoiling state to components in the region, with no signalling or state change being an option in case nothing is sending coil signals or receiving supercoiling signals. This implicitly means that our system works at the speed of supercoiling propagation. This is very fast in comparison to gene transcription, making the transcription rates hard to parameterise but also meaning that there is a time step during propagation when coiling has been caused but has not yet effected promoters.

Ignoring the bridges and including only the first fluorescent promoter, we can start to visualise the full circuit and its transitions. Let's define the circuit as:

$$circuit := G_{tetA} \,|\, S_1 \,|\, C_{red}| \, E_{red} \tag{11}$$

The G_{tetA} promoter updates the supercoiling region's state, S_1, which in turn updates the state of the fluorescent promoter, C_{red}.

We visualise this circuit as the circle shown in Figure 2. The G_{tetA} and C_{red} promoters on the circle are rectangles, indicating that they are DNA components. Inside the circle we place the diamond of the S_1 supercoiling region, the end of which is defined by ori, the origin of replication which acts as a supercoiling barrier in all plasmids. All the labels are updated by the transitions defined above; for the visualisation we also highlight the inside of the circle to indicate the part of the DNA that is negatively supercoiled.

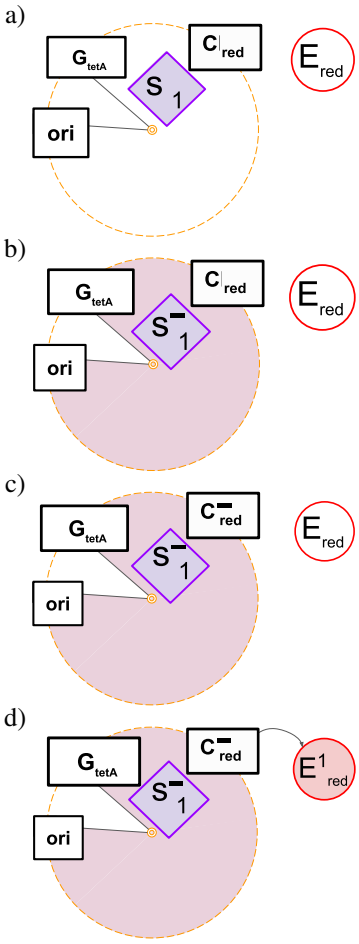

Figure 2: Visualisation of transitions from a–e for circuit with promoters G_{tetA} and C_{red}, supercoiling region, S_1 with Environments E_{red} and V. a) The inactive circuit b) Negatively supercoils the region c) Negatively supercoils the C_{red} promoter d) E_{red} starts to accumulate.

In Figure 2, we have components to receive the signals from the C promoters. These are modelled as environment components with a decay rate $d = 0$, which means the amount does not change except when more is detected:

$$E^i_{red} := (g^-_{red}\langle r\rangle | p_{red}(i)).E^{max(i+r,0)}_{red} + p_{red}(i).E^i_{red} + E^i_{red} \tag{12}$$

In this version of the Environments we add the ability to transmit the current protein level at the same time as it receives transcription signals from the promoters. This is done in parallel with the receipt using the "|" operator.

In the visualisation, Environments are circular elements outside the main circle; Environments for fluorescent proteins are shaded to match their label's colour.

Supercoiling Blocking Bridges

To control supercoiling in our example, we use a paired type of barrier, which we call a *bridge*. It needs two ends, BRA_n and BRC_n, which signal to up to 4 different supercoiling regions, referenced as S_{acw} the region anti-clockwise and S_{cw} of a particular bridge. They have the associated channels: s_{acw}, c_{cw}, as the bridge receives the supercoiling signal from the region anti-clockwise of the bridge point and repeats it as a coil signal to the region clockwise of the bridge point. This provides a proxy channel by which the supercoiling regions can communicate. They also listen for signals from the environment, $\bar{p_n}$ to "close" the bridge. If such a signal is received then a signal is exchanged between the ends of the bridge. This is done simultaneously so only if they both detect the signal from the environment and signals from each other can the bridge "close". If the bridge "closes" and is in state BRA'_n and BRC'_n then it constantly requires a continued signal to remain "closed"; otherwise it defaults to open.

$$BRA_n := \bar{s_{acw}}\langle u\rangle.c_{cw}(u).BRA_n + \bar{e}|b_a|\bar{b_c}.BRA'_n + BRA_n \tag{13}$$

$$BRC_n := \bar{s_{acw}}\langle u\rangle.c_{cw}(u).BRC_n + \bar{e}|b_c|\bar{b_a}.BRC'_n + BRC_n \tag{14}$$

$$BRA'_n := \bar{e}|b_a|\bar{b_c}.BRA'_n + BRA_n \tag{15}$$

$$BRC'_n := \bar{e}|b_c|\bar{b_a}.BRC'_n + BRC_n \tag{16}$$

This design of barriers and supercoiling regions interacting matches how G_{tetA} activates a supercoiling region and how components receive signals from supercoiling regions.

We consider the system of transitions for just the bridge. There are 6 components in this *bridge* system with the E_{LacI} component added:

$$E^i_{LacI} := \bar{g}\langle r\rangle.E^{max(i+r-d,0)}_{LacI} + p(i).E^{max(i-d,0)}_{LacI} + E^{max(i-d,0)}_{LacI} \tag{17}$$

$$bridge := E_{LacI}|BRA_{lac}|BRC_{lac}|S_1|S_2|S_3 \tag{18}$$

We assume that the S_1 region becomes negatively supercoiled. Then there are two possibilities: E^0_{LacI} is empty or too low to activate the bridge, else E^∞_{LacI} is full enough to activate the bridge. We start with the first situation:

$$\begin{aligned} &E^0_{LacI}|BRA_{lac}|BRC_{lac}|S^-_1|S_2|S_3 \\ &\quad\rightarrow E^0_{LacI}|c_2(-).BRA_{lac}|BRC_{lac}|S^-_1|S_2|S_3 \\ &\quad\rightarrow E^0_{LacI}|BRA_{lac}|BRC_{lac}|S^-_1|S^-_2|S_3 \\ &\quad\rightarrow E^0_{LacI}|BRA_{lac}|c_3(-).BRC_{lac}|S^-_1|S^-_2|S_3 \\ &\quad\rightarrow E^0_{LacI}|BRA_{lac}|BRC_{lac}|S^-_1|S^-_2|S^-_3 \end{aligned} \tag{19}$$

In Figure 3, we see in these transitions that the bridge is open. The BRA_{lac} transmits S_1 supercoiling regions over two transitions, the first it receives by $\bar{s_1}\langle u\rangle.c_2(u).BRA_{lac}$ the state transmitted by S_1 using $s_1(-).S^-_1$ and holds

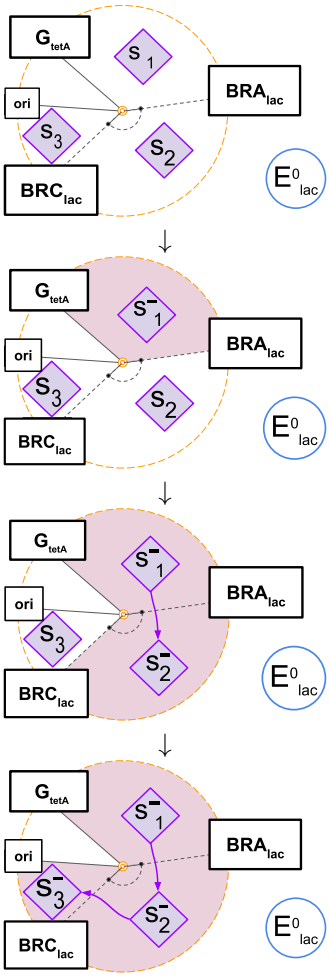

Figure 3: Visualisation and transitions for the circuit with a bridge and three supercoiling regions with environment node to represent the elements outside the DNA strand providing input into the bridge when it is inactive allowing supercoiling to propagate.

the data "-" in its new state. The second transition, $c_2(-).BRA_{lac}$, transmits the new state to S_2 which receives it with $\bar{c_2}\langle v\rangle.S_i^v$ and updates. In the third and fourth transmissions the same mechanism means S_2 transmits its new state to BRC_{lac} and it is received and held before being transmitted to S_3 which updates such that all the supercoiling regions are now negatively coiled. It is possible that BRC transmits a supercoiling signal before S_2 is updated, but this is fine as it will not change the state of S_3, this matches the stochastic nature of the biological circuits. The transmission of information is not entirely deterministic in its timings. It also accounts for if there is another barrier between BRA_{lac} and BRC_{lac}.

The alternative situation is E_{LacI} has contents sufficient to close the bridge. In this case we assume there is an infinite

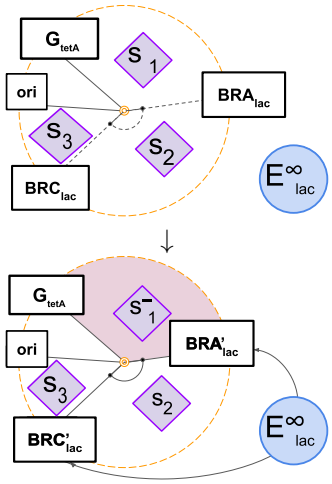

Figure 4: Visualisation and transitions for the circuit with a bridge, three supercoiling regions and environment node to represent the elements providing input into the bridge when it is active and preventing propagation of supercoiling.

supply of $LacI$ in the system:

$$E^{\infty}_{LacI}|BRA_{lac}|BRC_{lac}|S_1^-|S_2|S_3 \\ \rightarrow E^{\infty}_{LacI}|BRA'_{lac}|BRC'_{lac}|S_1^-|S_2|S_3 \tag{20}$$

Only one transition happens here, Figure 4, and the signals are all hidden as they occur simultaneously, checking the environment and both ends of the bridge in the transition. This changes the state of the bridge nodes and stops supercoiling propagation as the supercoiling regions are separated by the bridge barriers.

In Figures 4 and 3, we show the bridge as lines to the centre of the circle with both ends connected by an arc. These lines are either solid, when active, or dashed when inactive. When inactive the signalling between the supercoiling regions is displayed with an arrow to indicate the direction of transmission from one region to another. The signal goes through the open barrier and is visualised as going between the regions but is gone when the barrier is active. We include the G_{tetA} in the visualisation for completeness of the supercoiling system but leave out the promoters and fluorescents for now. These will be added back in for the final version of the circuit in the next section.

Example revisited

Finally, we are able to provide the full description of our initial motivating circuit, with all of its components, Figure 5. We provide the transitions of the system under different LacI conditions but only the visualisations of the static inactive version of the circuit and the two active versions after supercoiling has stabilised based on if LacI is present or not. The intermediate visualisations can be inferred from previous visualisations and the transitions.

In the transitions for the system without LacI the bridge passes on the signal to the next supercoiling region at the same time as the promoters are activated. This means there is only one additional transition for C_{blue} to signal the environment after its activation. We shorten the colour names initials in the following equations for brevity.

$$
\begin{aligned}
&G_{tetA}|E^0_{LacI}|S_1|C_r|E_r|BRA_{lac}|S_2|C_b|E_b|BRC_{lac}|S_3\\
&\rightarrow G_{tetA}|E^0_{LacI}|S^-_1|C_r|E_r|BRA_{lac}|S_2|C_b|E_b|BRC_{lac}|S_3\\
&\rightarrow G_{tetA}|E^0_{LacI}|S^-_1|C^-_r|E_r|BRA_{lac}|S^-_2|C_b|E_b|BRC_{lac}|S_3\\
&\rightarrow G_{tetA}|E^0_{LacI}|S^-_1|C^-_r|E^1_r|BRA_{lac}|S^-_2|C^-_b|E_b|BRC_{lac}|S^-_3\\
&\rightarrow G_{tetA}|E^0_{LacI}|S^-_1|C^-_r|E^2_r|BRA_{lac}|S^-_2|C^-_b|E^1_b|BRC_{lac}|S^-_3\\
&\rightarrow G_{tetA}|E^0_{LacI}|S^-_1|C^-_r|E^3_r|BRA_{lac}|S^-_2|C^-_b|E^2_b|BRC_{lac}|S^-_3
\end{aligned}
$$

The final state of this transition is referred to as a "steady state" i.e. as a state in which the system will remain (possibly with continued changes in the environment amounts but not effects). It is shown in Figure 5b.

The bridge reduces the active regions of the circuit. This reduces the transitions in the case of $LacI$ present in the environment. The number of transitions in the steady state system is the same as in the system without the bridge.

$$
\begin{aligned}
&G_{tetA}|E^\infty_{LacI}|S_1|C_r|E_r|BRA_{lac}|S_2|C_b|E_b|BRC_{lac}|S_3\\
&\rightarrow G_{tetA}|E^\infty_{LacI}|S^-_1|C_r|E_r|BRA'_{lac}|S_2|C_b|E_b|BRC'_{lac}|S_3\\
&\rightarrow G_{tetA}|E^\infty_{LacI}|S^-_1|C^-_r|E_r|BRA'_{lac}|S_2|C_b|E_b|BRC'_{lac}|S_3\\
&\rightarrow G_{tetA}|E^\infty_{LacI}|S^-_1|C^-_r|E^1_r|BRA'_{lac}|S_2|C_b|E_b|BRC'_{lac}|S_3\\
&\rightarrow G_{tetA}|E^\infty_{LacI}|S^-_1|C^-_r|E^2_r|BRA'_{lac}|S_2|C_b|E_b|BRC'_{lac}|S_3
\end{aligned}
$$

Similar to the bridge section, the circuit transitions with $LacI$ are shorter as the there is no propagation and delayed activation of the C promoters. This new "steady state" is shown in Figure 5c. We can see that the two systems produce two different environment outputs red in the first and red and $blue$ in the second, which indicates the inner workings of the DNA circuit and the level of output from the promoters. This is more helpful than the transcription rate itself. The transcription level can be deceptive as it is not the same as the protein in the system; it only acts as a signal for the production rather than its production itself. The model reflects this in the offset between the activation of the promoter (C^-_{red}) and the filling of the environment (E^1_{red}).

An implementation of this π-calculus as a plasmid circuit simulator is available: github.com/faulknerrainford/TORC

Discussion and Further Work

We have described a π-calculus model for plasmid DNA based gene circuits that includes the effects of supercoiling on gene expression. This allows us to model the Boolean logic systems and switches already used in plasmid circuits, and expand our capabilities to include supercoiling-based computation. One key advantage of π-calculus is that it simulates systems as a set of components, so can be expanded to simulate copies and further plasmids working within the same cell. This system is being tested as we engineer the model circuit described here in the wet lab. We will use the results of wet lab experiments and physical modelling based on theories of the formation of topological structures in DNA to include all supercoiling generated by all transcription in the plasmid. This approach could result in different or more dense computation in our plasmids as they exploit the extra process. As modelling becomes more accurate, production becomes more efficient. By including supercoiling in our modelling we improve the chance of building a correct and working plasmid in the wet lab.

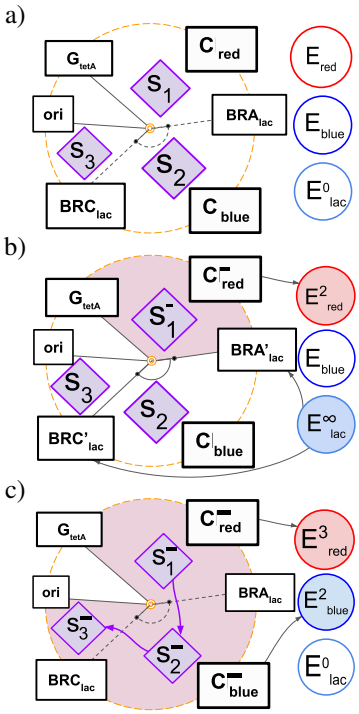

Figure 5: Full circuits from motivating example. a) the initial circuit before any supercoiling b) the stable state circuit in the presence of LacI with a closed bridge c) the stable state circuit in the absence of LacI with an open bridge

The π-calculus model is also presented as a visualisation designed to be intuitive to biologists and accessible to other users. The visualisation reflects the mathematics faithfully, and scales better than state transition figures. These visualisations will form the basis for an interface that allows biologists to work with our modelling. The instincts and insights of biologists who work with these complex systems are vital to their development. It is important as we build models and systems for generating plasmids as a source of computation that we keep biologists involved. Their ability to see problems beyond the scope of our models and to test the action of the plasmids, will result in better design processes for plasmid DNA circuits.

Acknowledgments. This work is funded by the TORC project (Supercoiling-driven gene control in synthetic DNA circuits), EPSRC-SFI grant EP/V027395/1.

References

Bi, C. and Benham, C. J. (2004). WebSIDD: server for predicting stress-induced duplex destabilized (SIDD) sites in superhelical DNA. *Bioinformatics*, 20(9):1477–1479.

Cardelli, L. (2005). Abstract machines of systems biology. In *Transactions on Computational Systems Biology III*, pages 145–168. Springer.

Curti, M., Degano, P., Priami, C., and Baldari, C. T. (2004). Modelling biochemical pathways through enhanced π-calculus. *Theor. Comput. Sci.*, 325(1):111–140.

Doronina, V. (2019). The beginner's guide to reading plasmid maps. https://bitesizebio.com/43119/the-beginners-guide-to-reading-plasmid-maps/. Accessed: 2023-3-8.

El Houdaigui, B., Forquet, R., Hindré, T., Schneider, D., Nasser, W., Reverchon, S., and Meyer, S. (2019). Bacterial genome architecture shapes global transcriptional regulation by DNA supercoiling. *Nucleic Acids Res.*, 47(11):5648–5657.

Fokkink, W. (1999). *Introduction to Process Algebra*. Springer Science & Business Media.

Friedland, A. E., Lu, T. K., Wang, X., Shi, D., Church, G., and Collins, J. J. (2009). Synthetic gene networks that count. *Science*, 324(5931):1199–1202.

John, M., Schulz, H.-J., Schumann, H., Uhrmacher, A. M., and Unger, A. (2013). Constructing and visualizing chemical reaction networks from pi-calculus models. *Form. Asp. Comput.*, 25(5):723–742.

Kim, J., White, K. S., and Winfree, E. (2006). Construction of an in vitro bistable circuit from synthetic transcriptional switches. *Mol. Syst. Biol.*, 2:68.

Kim, J. and Winfree, E. (2011). Synthetic in vitro transcriptional oscillators. *Mol. Syst. Biol.*, 7:465.

Kouzine, F., Sanford, S., Elisha-Feil, Z., and Levens, D. (2008). The functional response of upstream DNA to dynamic supercoiling in vivo. *Nat. Struct. Mol. Biol.*, 15(2):146–154.

Kwiatkowski, M. and Stark, I. (2008). The continuous π-Calculus: A process algebra for biochemical modelling. In *Computational Methods in Systems Biology*, pages 103–122. Springer.

Lakin, M. R., Youssef, S., Cardelli, L., and Phillips, A. (2012). Abstractions for DNA circuit design. *J. R. Soc. Interface*, 9(68):470–486.

Liu, L. F. and Wang, J. C. (1987). Supercoiling of the DNA template during transcription. *Proc. Natl. Acad. Sci. U. S. A.*, 84(20):7024–7027.

Lodge, J. K., Kazic, T., and Berg, D. E. (1989). Formation of supercoiling domains in plasmid pBR322. *J. Bacteriol.*, 171(4):2181–2187.

Lou, C., Liu, X., Ni, M., Huang, Y., Huang, Q., Huang, L., Jiang, L., Lu, D., Wang, M., Liu, C., Chen, D., Chen, C., Chen, X., Yang, L., Ma, H., Chen, J., and Ouyang, Q. (2010). Synthesizing a novel genetic sequential logic circuit: a push-on push-off switch. *Mol. Syst. Biol.*, 6(1):350.

McCaskill, J. S. and Niemann, U. (2001). Graph replacement chemistry for DNA processing. In *DNA Computing*, pages 103–116. Springer.

Parrow (2001). An introduction to the π-calculus. *Handbook of process algebra*.

Phillips, A. and Cardelli, L. (2009). A programming language for composable DNA circuits. *J. R. Soc. Interface*, 6 Suppl 4:S419–36.

Regev, A. and Shapiro, E. (2004). The π-calculus as an abstraction for biomolecular systems. In Ciobanu, G. and Rozenberg, G., editors, *Modelling in Molecular Biology*, pages 219–266. Springer.

Regev, A., Silverman, W., and Shapiro, E. (2000). Representation and simulation of biochemical processes using the π-calculus process algebra. In *Biocomputing 2001*, pages 459–470. WORLD SCIENTIFIC.

Sumpter, D. J., Blanchard, G. B., and Broomhead, D. S. (2001). Ants and agents: a process algebra approach to modelling ant colony behaviour. *Bull. Math. Biol.*, 63(5):951–980.

Tabor, J. J., Salis, H. M., Simpson, Z. B., Chevalier, A. A., Levskaya, A., Marcotte, E. M., Voigt, C. A., and Ellington, A. D. (2009). A synthetic genetic edge detection program. *Cell*, 137(7):1272–1281.

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.

Wang, H., Kaloper, M., and Benham, C. J. (2006). SIDDBASE: a database containing the stress-induced DNA duplex destabilization (SIDD) profiles of complete microbial genomes. *Nucleic Acids Res.*, 34(Database issue):D373–8.

Yordanov, B., Kim, J., Petersen, R. L., Shudy, A., Kulkarni, V. V., and Phillips, A. (2014). Computational design of nucleic acid feedback control circuits. *ACS Synth. Biol.*, 3(8):600–616.