LU TP 18-28
September 13, 2018

Identifying Cell Reprogramming Roadblocks by Modelling Induced Pluripotency with Boolean Networks and Dynamical Systems

Johannes Isaksson

Department of Astronomy and Theoretical Physics, Lund University

Master thesis supervised by Victor Olariu

LUND
UNIVERSITY

Abstract

The generation of induced pluripotent stem (iPS) cells from differentiated cells is a process of great scientific interest due to the medical potential of such cells. This process (called 'reprogramming' of cells) is however notoriously inefficient and still not very well understood. Therefore studies that can help us understand the interactions and mechanisms governing the reprogramming process are of great importance.

In part one of this project, we develop a Boolean network that is able to emulate the full transition from mouse embryonic fibroblast (MEF) cells to iPS cells and show that it can be used to predict real knockdown behavior with an overall accuracy of 83%. We also establish a new way to accurately measure *in silico* reprogramming efficiency of Boolean models.

Part two of this project involves the construction of a minimal dynamical systems model for gene regulatory networks that is able to simulate the reprogramming process. We show that this model accurately replicates certain key features in the cell reprogramming process and predicts that factors exhibiting specific expression dynamics act as roadblocks for cell reprogramming.

Overall, the results of the two parts of this project provide valuable insights on the cell reprogramming process and help identify factors acting as roadblocks.

Populärvetenskaplig sammanfattning

Människokroppen är uppbyggd av olika typer av celler som exempelvis hudceller, nervceller och fettceller. Generellt sett är dessa celler utformade för en väldigt specifik uppgift såsom att utföra arbete (exempelvis muskelceller) eller att förse kroppen med struktur (exempelvis benceller). Det finns en väldigt spektakulär typ av cell – så kallade stamceller – vars specifika syfte är att omvandlas till andra celltyper och det är via dessa celler som kroppen har möjlighet att skapa sina olika celler. Processen som beskriver stamcellers övergång till andra celltyper kallas differentiering och är mycket intressant från ett medicinskt perspektiv eftersom den ger oss en viktig inblick i hur människokroppen fungerar. Nyligen har forskare upptäckt en metod för att odla fram stamceller från differentierade celler i en process som har blivit benämnd "omprogrammering" av celler. Genom att omprogrammera celler kan forskare alltså odla fram stamceller från exempelvis hudceller för att sedan använda stamcellerna i medicinska applikationer såsom stamcellsterapier som har visat sig framgångsrika vid behandling av exempelvis ALS eller Alzheimers sjukdom.

Ett av de största problemen med att odla fram stamceller via omprogrammering i en labbmiljö är den långsamma processen och den väldigt låga effektiviteten. Dessa problem orsakas till stora delar av så kallade "väghinder" som är ett samlingsnamn för olika mekanismer som kan sakta ner eller på olika sätt förhindra processen från att äga rum. Exempel på väghinder kan vara att "slå av" gener som inte har med stamceller att göra eller motsvarande att "slå på" gener som är relaterade till stamceller. Att identifiera dessa väghinder och på så sätt underlätta framställandet av stamceller är en av de största utmaningarna för forskningen inom detta område.

Detta projekt behandlar olika metoder för att hantera och upptäcka dessa väghinder. I projektet utforskas olika matematiska modeller för omprogrammering som sedan används för att identifiera möjliga väghinder i form av specifika geners beteende. Enligt organisationen Alzheimer's Disease International lider för närvarande omkring 50 miljoner människor världen över av någon typ av demenssjukdom och andelen drabbade förväntas öka de närmaste 30 åren. Parkinsons sjukdom, leukemi och ALS är andra exempel på potentiella behandlingsområden där stamcellsterapier har mycket potential. I det långa loppet är målet med forskningen inom detta område att på sikt utrota dessa folksjukdomar och förhoppningsvis kan arbetet i detta projekt vara ett steg i rätt riktning.

Contents

List of Figures

List of Tables

1 Introduction

Takahashi and Yamanaka demonstrated in 2006 that induced pluripotent stem (iPS) cells can be cultivated from mouse embryonic fibroblast (MEF) cells in a process called reprogramming [1]. This process has since then been studied extensively due to the medical potential of stem cells in fields such as regenerative medicine [2], drug development [3], stem-cell therapies for treating e.g. Alzheimer's disease [4] etc. In recent years, high-throughput technologies have made it possible to gather large quantities of data effectively, which has surged an increase of published studies [5]. However, our understanding of the control process behind induced pluripotency and the generation of iPS is lagging behind.

A major issue with *in vitro* cell reprogramming is the extremely low efficiency and slow kinetics of the process, often attributed to reprogramming obstacles collectively known as 'roadblocks'. Identifying potential roadblocks is critical for understanding the process and improving its efficiency.

Fully understanding the process of cell reprogramming involves describing and modelling the interactions and behaviors of the relevant genes inside the cell by constructing its gene regulatory network (GRN). Two of the most common methods for this are Boolean network (BN) models and dynamical systems models. BNs are mathematical structures of nodes (representing genes) [6] where the value of each node takes on a value of either 1 (ON) or 0 (OFF) depending on whether or not the genes are being expressed. The interactions between genes in a BN are modelled with Boolean functions. The dynamical systems approach involves setting up a system of differential equations describing gene expression level dynamics over time. By solving and evaluating these equations one can then infer behaviors and characteristics of the GNR.

This project consists of two parts. In part one, the reprogramming process is modelled by developing a BN and training it to reproduce experimentally observed constraints. To help with this, a novel way of establishing the level of pluripotency of cells *in silico* is developed in order to measure the efficiency of cell reprogramming in simulations. Predictions of potential roadblocking genes in the BN model are validated with respect to experimental observations, showing an accuracy of 83%.

Secondly, we construct a simplified dynamical systems network which is used to model the generation of iPS cells. Stochastic simulations of the simplified model reveals that certain genes with a specific expression pattern exhibit roadblock-like behaviors.

2 Background

2.1 Gene Regulation

The central dogma of molecular biology describes the flow of genetic information within an organism or biological system. Figure 1 shows a schematic illustration of this biological process, in which the phrase *gene regulation* is a collective term for the transcription and translation processes. An essential factor in this flow of information is the existence of

transcription factors (TFs), which are proteins that control transcription of specific genes by either turning transcription on (activating) or off (repressing) by binding or not binding to corresponding binding sites on the DNA. It is often convenient to model the transcription and translation processes as a single process with a single equation in order to simplify the mathematical model.

The biological processes governing gene regulation are extremely complex and very hard to fully understand because so many factors are at play. Therefore, approximations and assumptions are necessary. For example, gene regulation is often modelled by a closed system, even though this is certainly not the case for real biological systems.

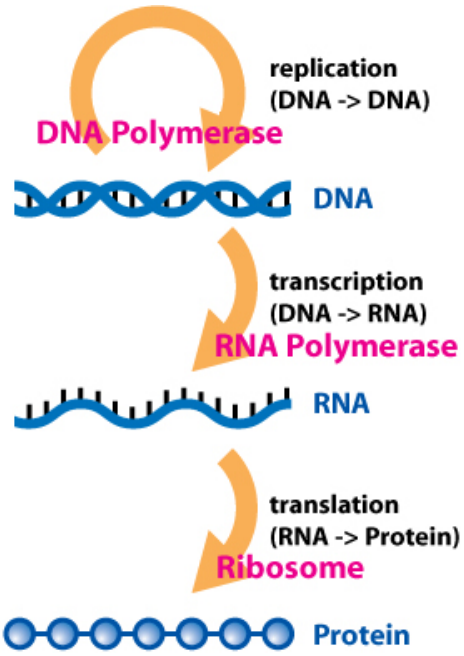

Figure 1: Illustration of the flow of genetic information as stated by the central dogma of molecular biology. Replication: DNA passes coded information by replicating via DNA Polymerase. Transcription: coded information is passed on via RNA by RNA synthesis governed by RNA polymerase. Translation: messenger RNA carries information on to ribosomes where protein synthesis takes place. Proteins carrys no information themselves but helps perform almost all biological tasks.

2.1.1 Gene Regulatory Networks

A gene regulatory network (GRN) is a set of molecular regulators controlling the gene expression in cells and organisms. The expression of the genes is embodied by the presence of their corresponding RNA, which in turn carries the information 'coded' within them required for synthesis of their corresponding proteins. Proteins and RNAs themselves control most of a cells behaviors and mechanisms, which makes the study of regulatory networks paramount in almost all areas of biology.

Component genes in a GRN are conventionally expressed as nodes in a directed graph. In such a graph the regulatory interactions are represented by its edges – for instance an activating relation of gene B by gene A is represented by a directed arrow from A to B. A repressing relation is typically represented by a directed blunted line.

Mathematical models for GRNs were proposed already in 1969 by Britten & Davidson [8] as well as Kaufmann [9] in the same year. The role for these mathematical models include [10]:

- Describing genetic regulations at a system level
- Enabling simulation of network behavior
- Predicting new structures and relationships
- Making it possible to analyze or intervene in the network through signal processing

The task of fully understanding genetic interactions is hard and countless studies have been conducted in the matter. Several methods have been proposed to model gene interactions, including coupled ordinary differential equations [11], Boolean networks [12], stochastic gene networks [13], master equations [14], and all with their respective variations.

Boolean networks are one of the most extensively used frameworks due to their simple yet powerful nature. Moreover, they have been proven to successfully describe real gene regulatory relations such as the drosophilia segment polarity network [15].

2.2 Boolean Networks

A Boolean network (BN) is a mathematical structure $\beta(V, E)$, built on a gene regulatory network with V vertices (nodes) representing the genes and E directed edges representing their interactions. At any given state or point in time each node has a value of either 0 or 1 i.e. on or off. The value of each node in the next state or point in time is determined by the values of the parent nodes (incident-directed nodes, also called the predictor set) via Boolean algebraic functions:

$$f_i : \{0,1\}^{p_i} \longrightarrow \{0,1\} \tag{2.1}$$

for node i with p_i parents. As such, each node has a corresponding truth table associated to it which completely determines its Boolean map f_i.

One of the challenges of Boolean networks is determining the underlying Boolean functions (i.e. filling in the truth table outputs) for each node – a process often called 'training' the network. Many different types of Boolean network models have been developed for gene regulation, including the general model [16], the AND/OR model [12], the strong-inhibition model [17] and several different stochastic models [13][18].

A state in a Boolean network with n nodes is a vector of gene values $(x_1, ..., x_n)$ and the total number of states in such a network is accordingly 2^n. Furthermore, the next state of

a network can be updated either synchronously or asynchronously depending on whether the nodes are updated all at the same time or not. If the output of a state is itself, i.e.

$$f(x_1(t), \ldots, x_n(t)) = f(x_1(t+1), \ldots, x_n(t+1)) \tag{2.2}$$

that state is a single-state attractor. There are also attractor cycles which are irreducible subsets of total state-space with the property that they cannot be exited once entered. An attractors basin of attraction (or simply basin) is the set of all states which inevitably end up in that attractor.

Note that the output of a BN is unique in the synchronous update, but not in the asynchronous update. A consequence of this is that the attractor structures may differ between the cases. To see this, consider a three-node BN with synchronous updates and an attractor cycle 111→001→111→ .... This cycle cannot exist in the asynchronous case because it requires multiple flips at the same time. On the other hand, all state-transitions that can be made in the asynchronous case can also be made in the synchronous case. This means that the single-state attractors are inevitably identical in both cases.

Studying basins of attraction is a staple task in the study of BNs because they have been associated with cellular phenotypes in living organisms [19]. Furthermore, the size of the basins are associated with the probability of a cell having a certain phenotype in real organisms.

2.2.1 Boolean Maps

The Boolean algebraic function $f_i : \{0,1\}^{p_i} \longrightarrow \{0,1\}$ is a Boolean map from $x \in \{0,1\}^{p_i}$ to $x \in \{0,1\}$, which determines the state for node i with p_i parents in the next instance in time by $x_i(t+1) = f_i(x_1(t), \ldots, x_{p_i}(t))$. Every node in a GNR has its own parents and Boolean function f. Conveniently, Boolean functions can be expressed in terms of logical operators AND, OR, NOT etc., which can often elucidate relationships between variables. For example, the logic gate AND activates only when variable A *and* variable B are active. Its truth table is shown below

Table 1: Truth table of the logic AND gate.

A	B	AND(A, B)
0	0	0
1	0	0
0	1	0
1	1	1

When several inputs are present and compositions of Boolean functions are considered, it gets increasingly complex because the internal order starts to matter. For example,

AND(A,OR(B,C)) and AND(C,OR(A,B)) will not produce the same truth table. Similarly, the order of the operators also matter such that OR(A,AND(B,C)) is not equal to AND(A,OR(B,C)) etc.

Part of the success of BNs in biological modelling comes from the parallel that can be drawn directly from logical operators to genetic regulation. If for instance a repressing molecule binds to a receptor the describing logic is a NOT gate because transcription can only be active when the molecule is not bound (NOT(A)=1 if and only if A=0). Similarly, if transcription is active if either molecule A or molecule B is bound, the corresponding logic operator is OR(A,B). This argument can be extended to other logic gates and be applied to several inputs as well.

Sometimes it is of interest to make the discretization of values in the BN to more than two levels. A discretization to k levels will transform the mapping to $f_i : \{0, \ldots, k\}^{p_i} \longrightarrow \{0, \ldots, k\}$ [10]. The benefit of this is the increased resolution of the model but it comes at the heavy cost of computational efficiency; a k-level BN will have k^n different states. With such a discretization, the definition of the logical operators are updated to be consistent with higher values. The three most common operators are defined as $AND(x_1, \ldots, x_n) := min(x_1, \ldots, x_n)$, $OR(x_1, \ldots, x_n) := max(1, \ldots, n)$ and $NOT(x) := n - 1 - x$ in a multi-leveled system.

2.2.2 Probabilistic Boolean Networks and Stochastic Modelling

Probabilistic Boolean networks (PBNs) have been developed to account for the self-evident stochasticity in real-life molecular and genetic regulations. Gene regulation in real organisms is not only subject to cellular noise but is also affected by external changes as well as other genetic networks since they are not closed systems. A PBN is a composition of several BNs in which each constituent BN is associated with a network selection probability c_i. At any given time only one BN determines the next state. Every time instant, there is a switching probability q and every BN has its corresponding selection probability c_i to be chosen as the BN for the next update. Additionally, one may include a rate of random gene perturbation p.

Probabilistic Boolean models are homogeneous Markov chains (MCs). This is perhaps easier realized when thinking of the PBN in terms of its state transition probability matrix T in which each state (row) has certain probabilities to transition to other states (columns). The condition $\sum_{j=1}^{2^n} T_{ij} = 1, \forall i$ for probabilities in the transition probability matrix together with the fact that transitions are independent of previously visited states implies a MC. Since the transition matrix is the same in each time-step, the chain is homogeneous. The MCs are also ergodic because they have a finite and irreducible state space with aperiodic states (in fact, gene perturbation alone is a sufficient condition for this to be the case). Importantly, a consequence of this is that PBNs possess *steady-state distributions*, which are unique long term behaviors independent of their initial states. The study of steady-state probabilities is one of the most significant applications of PBNs for gene regulatory networks [20].

PBNs will be used in this project in order to assess the stochastic behavior of the BN

model.

2.2.3 Simulating Boolean Network Dynamics

There are two ways to simulate the dynamic time evolution of a BN in the synchronous case. First, given a state $\mathbf{x}$, one can look up the Boolean functions for each node directly and compute its output with respect to the parent-nodes. In effect, this method revolves around finding the corresponding row in each node's truth table and saving the output value.

The other method, sometimes called the matrix-based method, is more effective when simulation time is very long (typically $t \gg 2^n$). It utilizes the state transition matrix (TM) of the BN, which holds the output state of each state in a 1×2^n array for lexicographically ordered states $x \in \{0, \ldots, 2^n\}$. However, this method requires computation of the TM, which is not always practically feasible, especially for large BNs.

When a perturbation rate is introduced and/or there is a switching probability between BNs in a PBN, the switch/perturbation occurs before matrix/table look-up. In the matrix-based method, this enables computation of the time evolution of a PBN without having to calculate its $2^n \times 2^n$ transition probability matrix. Instead, if perturbation occurs, one converts the integer state $x \in \{0, \ldots, 2^n\}$ to its corresponding Boolean state $\mathbf{x} \in \{0, 1\}^n$, makes the flip/flips, then makes the conversion back again.

One issue with modelling transcriptional dynamics with a BN is the choice of gene perturbation rate p and network switching probability q. Previous studies have proposed values of $p = 0.01\% - 0.5\%$ and $q = 1\% - 20\%$ in order to reflect the rarity of random gene perturbation and change of biological context [10].

2.2.4 Markov Chain Analysis

With the Markov chain (MC) property of PBNs it is possible to conduct theoretical calculations of the long term behavior of networks. The transition probability matrix T of a MC portray the transition probabilities from each state to every other state and is an important item in Markov chain analysis. A MC with n genes possesses a stationary distribution π if the probability distribution $\pi = (\pi_1, \ldots, \pi_n)$ is such that

$$\pi = \pi T \tag{2.3}$$

for its transition matrix T and constraint $\sum_i \pi_i = 1$. This implies $\pi = \pi T^n \ \forall n$, in which T^n is the n-step probability matrix of the MC, i.e. its (i, j)-th element is the probability to transition from state i to state j in n steps [10]. In other words, after starting from the i-th state with probability π_i, the probability of finding the MC in any state j after an arbitrary number of steps is always π_j. The distribution π^* is called a steady-state distribution if for all n and any initial distribution π the following holds:

$$\pi^* = \lim_{n \to \infty} \pi T^n \tag{2.4}$$

In practice, this means that the probability of being in any state j after a long time is always π_j^* *regardless* of the initial state. An ergodic MC will always posses a steady-state distribution [10].

As previously mentioned, the steady state distributions of MCs are an important asset in the study of GRNs because they are associated with cellular phenotypes in real organisms. Finding and analyzing them may thus add valuable insights to the behaviour and characteristics of such a system.

2.2.5 Computational Methods for Finding the Steady State Distribution

The analytical method for obtaining the steady state distribution above is reduced to finding the solution to a system of linear equation with constraint $\sum_i \pi_i = 1$. The following algorithm, called the power method, can compute the steady state distribution π^* for the full state space [10]:

Input: Initial distribution $\pi^{(0)} = (\pi_1^{(0)}, \ldots, \pi_n^{(0)})$, transition probability matrix T, error tolerance δ^*.
Output: Steady-state distribution π^*.
k=0;
while $\delta < \delta^*$ **do**
  Compute $\pi^{(k+1)} = \pi^{(k)} \cdot T$;
  Set $\delta = ||\pi^{(k+1)} - \pi^{(k)}||$;
  Set $k = k + 1$;
end
Set $\pi^* = \pi$;

Algorithm 1: Power method for obtaining the steady-state distribution. This method iterates computations of the distribution π following the definition in Equation 2.4.

The problem with the method above is that it involves computation with large matrices T of size $2^n \times 2^n$. Some authors have suggested approximation methods for computing T [22], but the problem remains that sometimes T is even too large to be stored in a conventional computer.

An alternative method for computing the steady-state distribution is with Monte-Carlo simulations. These methods generate long time series of the PBN such that the frequency of states approaches the steady-state distribution. In general, one needs to simulate at least $10 \cdot 2^n \cdot p^{-1}$ steps in order to reach the steady state distribution [10], which is typically far too many to be run in a convenient time frame. Therefore, it is customary to utilize one of many ways to test the convergence of an empirical distribution [23, 24] and thence approximate the steady-state distribution with the empirical distribution.

2.3 Biochemical Dynamics and Kinetics

To make a quantitative model of a biochemical network, one has to define a system of mathematical functions explicitly describing the interactions of its variables. The variables represent molecular concentrations of for instance mRNA, enzymes, proteins, complexes etc. There are a number of different ways one can go about defining the characteristics of the system; variables can be discrete or continuous, dynamics can be deterministic or stochastic, update rules can depend on internal or external factors and so on.

When modeling reaction kinetics, particularly transcriptional dynamics, there are a few different standard approaches. Excluding stochastic and Boolean models, the two main methods are the Michaelis-Menten [27] (with Hill formalism [28]) and the Shea-Ackers approach [29]. The approaches use different methodologies to establish the dynamics but the resulting governing equations of the systems end up being equivalent, as they necessarily should.

In a dynamical systems approach, the equations governing the state variables of the system are sets of ordinary differential equations (ODEs) describing the evolution of the variables over time. The set of variables then defines the state of the system and the system is said to be in a stationary state if the variables are in equilibrium (i.e. their time-derivatives are zero), or have reached a stationary limit cycle (i.e. oscillating state). Just as in any other dynamical system, stationary states can be stable or unstable depending on if they attract or repel nearby perturbations. A benefit of defining the system dynamics in this way is that the future behaviour of the system is unique and entirely known, given the parameters and external factors do not change. Additionally, the resolution of a dynamical systems model is higher than that of a Boolean networks model because the values of the variables are not discretized.

2.3.1 The Shea-Ackers Formalism

The Shea & Ackers approach to modelling TF binding states is based on a statistical physics view. All possible binding states (bound or unbound for all possible TFs and potential complexes) definey a partition function and the transcription rate is proportional to the probability of having an active transcribing state, i.e. $P_{transcription} = Z_{active}/Z_{total}$ where Z is the partition function, defined by

$$Z = \sum_{\sigma_1 \ldots \sigma_n} \prod_{i}^{n} [TF_i]^{\sigma_i} e^{-\Delta G_\sigma / k_b T}. \tag{2.5}$$

In this equation, the probability of having transcription factor i bound is proportional to its concentration $[TF_i]$. Each state has an associated free energy ΔG_σ and the states can either be bound ($\sigma_i = 1$) or unbound ($\sigma_i = 0$). k_b is the Boltzmann constant and T is the temperature. Z is conventionally normalized such that the statistical weight of the state where nothing is bound equals one. The products in this partition function accounts for the different combinations for multiple TFs being bound at the same time. For instance, if TF_1

and TF_2 being bound simultaneously is a possible state, the product of their concentrations will be one term in the partition function.

The Shea-Ackers formalism assumes a homogeneous well-stirred solution such that the concentrations are functions of time and not space. Additionally, the reactions are assumed to occur in a deterministic manner, and the number of reactants is assumed to be of order of magnitude much larger than one [30].

If the possibility for TFs to bind at multiple sites at the DNA is introduced, equation 2.5 becomes more responsive to changes in substrate concentrations, which is often evident in real biological systems. Mathematically, this is modelled by the introduction of an exponent, δ_i, in the concentration factors, called the Hill coefficient. By letting $\beta_i = e^{-\Delta G_\sigma / k_b T}$ the equation becomes (excluding complex formation):

$$\frac{d[TF_i]}{dt} = \frac{\sum_{i_{active}} \beta_i [TF_i]^{\delta_i}}{Z} \tag{2.6}$$

The Hill coefficient reflects the degree of cooperativity between binding molecules such that a high cooperativity results in a steep response. The constants β_i are binding affinities for the TFs which represent the rate of binding. To make the dynamics account for decay, a concentration dependent decay term $\gamma[TF_i]$ is subtracted from the equation and the final model equation is

$$\frac{d[TF_i]}{dt} = \frac{\sum_{i_{active}} \beta_i [TF_i]^{\delta_i}}{Z} - \gamma[TF_i] \tag{2.7}$$

The challenge here is to approximate the model parameters from experimental data. It is also clear that model complexity increases quickly with system size, making the dynamical systems approach unsuitable for larger networks.

2.4 Stem Cells, Differentiation and Reprogramming of Cells

Stem cells are a specific type of cells which are characterized by their ability to self-renew and their pluripotency. Their pluripotent property means that they have the potential to differentiate, which means that they have the ability to transform into other cell types. Self-renewal ensures that they can replicate indefinitely while still maintaining an undifferentiated state. In adult organisms, stem cells play a crucial role in e.g. healing and regeneration, which makes the study of stem cells critical for its potential applications in medicine.

Initially discovered by Yamanaka as recently as 2006 (for which Yamanaka was awarded the Nobel prize in medicine in 2012), artificial reprogramming of cells is a process where induced pluripotent stem cells (iPS cells) are grown from mature cells such as fibroblast cells [1]. Yamanaka demonstrated that this could be done with mice cells by exposing (conventionally called overexpressing) the cells to the exogenous transcription factors Oct4, Sox2, Klf4, and c-Myc. Since its discovery, cell reprogramming has been investigated further and several ways to enhance its efficiency have been revealed, but the underlying mechanisms of the process are still not very well understood.

There is, however, a technical issue in the reprogramming process regarding the assessment of a pluripotent stem cell. An iPS cell is by definition a cell which has re-gained pluripotency but the problem is that there is no strict way of examining the pluripotent property of the cells *in vitro*. Experimentally, researchers typically look at the morphology of the cells and their ability to replicate, neither of which is present in any computational model.

This project will be reliant on data from experiments (see Section 2.5) conducted with certain pluripotency markers such as the Nanog green fluorescent protein (GFP) reporter and the cell surface markers CD44 and ICAM1. It has been shown that pluripotent stem cells have a high clonogenicity of Nanog GFP, which is transiently increased from virually zero in the fibroblast stage. The Nanog-GFP colony forming potential (CFP) thus gives an indication of how pluripotent the cells are. Additionally, the cell surface markers CD44 and ICAM1 have been shown to follow an ordered sequence of changes corresponding to cell-stage transitions when reprogramming [32].

Maintaining pluripotency of cells *in vitro* requires culture environments which support stable self-renewal. Currently optimized conditions include different combinations of cytokine leukemia inhibitory factor (LIF) and two selective inhibitors CH and PD (collectively known as 2i) [31]. In these conditions, cells homogeneously express pluripotency-related factors and show no signs of differentiation. These three factors are therefore an integral part in the reprogramming process.

2.4.1 Modelling Pluripotency and Overexpression

As mentioned in the previous section, neither the morphology of cells nor their ability to replicate are typically modelled when researching pluripotency of cells *in silico*. Therefore, previous researchers have instead looked at i.e. the expression levels of key pluripotency-related factors such as Oct4 and Sox2 in order to establish pluripotency. One may then assume that cells are pluripotent only if both Oct4 and Sox2 are expressed simultaneously [31]. A problem with this is that there are dozens of pluripotency-related factors and not all of them need to be present in a cell for the cell to actually be pluripotent.

This project will explore a novel way of looking at pluripotency of cells *in silico* akin to the aforementioned Nanog-GFP CFP *in vitro*. Details of the procedures behind this will be covered in the methods of section 4.

When it comes to modelling the overexpression of TFs the approach is different depending on which framework is being used. In a dynamical systems framework one can simply introduce an overexpression term α to the dynamical equations such that Equation 2.7 becomes

$$\frac{d[TF_i]}{dt} = \frac{\sum_{i_{active}} \beta_i [TF_i]^{\delta_i}}{Z} - \gamma[TF_i] + \alpha_i \tag{2.8}$$

where α_i is zero for non over-expressed factors and needs to be determined for the Yamanaka factors.

For Boolean networks there are two possibilities: either force the expression levels of the overexpressed factors to a high expression state, or add new overexpression nodes in the network for the overexpressed genes. The advantage of the latter is that it enables distinction between the exogenous and endogenous transcription factors.

2.5 Experimental Background

This section covers two studies of GNRs that have provided experimental data that will be used in this project. Results from both studies will be used as training data when constructing the Boolean network model in the first part of this thesis. Additionally, gene time-series data from the first study will be used as target values when constructing the dynamical systems network in the second part.

The first study in particular proved extra useful because it studied the exact same process that was modelled in this project, namely the generation of iPS cells from mouse embryonic fibroblast (MEF) cells by over-expressing the reprogramming factors Oct4, Sox2 and Klf4. The second study only considered cells already in the iPS state, but nevertheless provoided useful gene expression data.

2.5.1 Reprogramming Stage Transitions and Time-Series

This section covers results from the publication by O'Malley et al. from 2013 [32], which provided valuable experimental insights and data. In their study they demonstrated that reprogramming follows an orderly sequence of stage transitions marked by changes in cell surface markers CD44 and ICAM1, and a Nanog-GFP reporter. They carried out RNA-sequencing which suggested a binary classification of behaviors of pluripotency-related genes in the reprogramming process: rapid ('early' genes) or gradual ('late' genes) up-regulation. They also discovered transient up-regulation patterns for epidermis-related genes which suggests that reprogramming is not simply a reversal of the differentiation process.

Their results were incorporated in this project in two major ways. First, the different stages in the reprogramming process were applied as attractors in the Boolean Networks constructed in this thesis (see Section 4). Secondly, the gene expression patterns in the different stages were used as time-series data for the optimization of parameters in the dynamical system approach. It is important to understand how their results were produced in order make accurate implementations of the results from a systems biology perspective.

The different stages between MEF and iPS are called 1NG-, 1NG+, 2NG-, 2NG+, 3NG- and 3NG+, where the 'NG-/+' stands for whether the stage has high or low Nanog-GFP expression. The integer 1-3 indicates which *gate* the cells are in. The different gates are distinguished by the expression levels of ICAM1 and CD44:

1. Gate 1: low expression of ICAM1, high expression of CD44
2. Gate 2: low expression of ICAM1, low expression of CD44

3. Gate 3: high expression of ICAM1, low expression of CD44

Cells tend to follow the transition MEF→ Gate1→ Gate2→ Gate3→ iPS when being reprogrammed. The whole process from MEF to iPS can be seen in Figure 2. Cells that have advanced to a new stage after a few days are sorted and re-plated repeatedly until an iPS colony is achieved, which effectively disposes of the cells that are lagging behind. A consequence of this is that there is no real time-series data. Nevertheless, the expression of the pluripotency genes in the reprogramming stages will be used as a time-series approximation since they still follow chronological order. Without sorting it takes more than 20 days to get any iPS-like cells at all, and larger iPS colonies are rarely ever formed.

An excerpt from the experimental expression levels in the different stages is shown in Figure 3. The figure shows ensemble averages of expression levels of genes classified into two groups: Early/E (Oct4, Sall4 and Tcfp1l1), and Late/L (Esrrb, Gbx2, Klf2, klf5, Nanog and Sox2). Stat3 and Klf4 are displayed as a separate entry because they do not quite follow this classification.

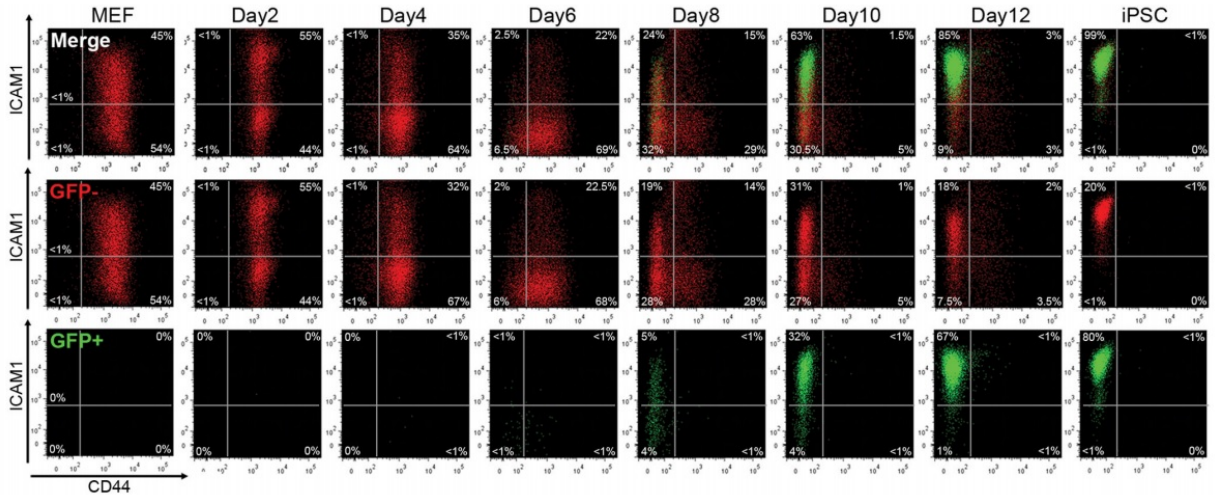

Figure 2: Experimental results from [32] illustrating the path from MEF cells to iPS cells and the intermediate transitions from 1NG- to 3NG+ in between. The bottom-right areas (Cd44+/Icam1-) are the first gate , the bottom-left areas (Cd44-/Icam1-) are the second gate, and the upper-left areas (Cd44-/Icam1+) are the third gate. The green results of the bottom row shows the cells Nanog-GFP colony forming potential. Note that NG+ cells (i.e. cells with high Nanog-GFP CFP) do not appear until day 8. The cells are gate-sorted and replated several times in the reprogramming process.

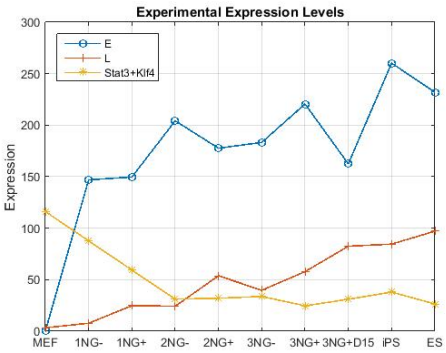

Figure 3: Selection of experimental expression levels of genes in different reprogramming stages extracted from [32]. This figure was created by averaging the expression levels of all early (E; Oct4, Sall4, Tcfp2l1) and late (L; Esrrb, Gbx2, Klf2, Klf5, Nanog, Sox2) genes together in two different series. It also displays Stat3 and Klf4 as a separate entry because of their unique expression pattern.

2.5.2 Knockdown Studies and Culture Perturbations

This section covers the experimental findings of Dunn et al. [31]. In their publication they tried to develop a minimal set of network interactions that could describe the propagation of embryonic cell identity. They, too, employed a Boolean networks approach, but their methods were different from the ones in this project. For this, they used an array of experimental knockdown studies and culture perturbations as seen in Figure 4. Gene knockdown (KD) is an experimental technique where the expression level of a specific gene is reduced by blocking its transcription. Such studies produce valuable information of the effects of KD, such as the role of specific genes or their relative importance. Different cultures in this context refers to the different combinations of external factors cells are cultivated in. These are the top three rows of Figure 4, i.e. different combinations of LIF, CH and PD.

The main realization here is that right columns in Figure 4 correspond to experimentally stable stem-cell states which should hence be present in the computational model as stationary states. Therefore, these results will be used as training data for the BN model in this project (see section 4.1.1).

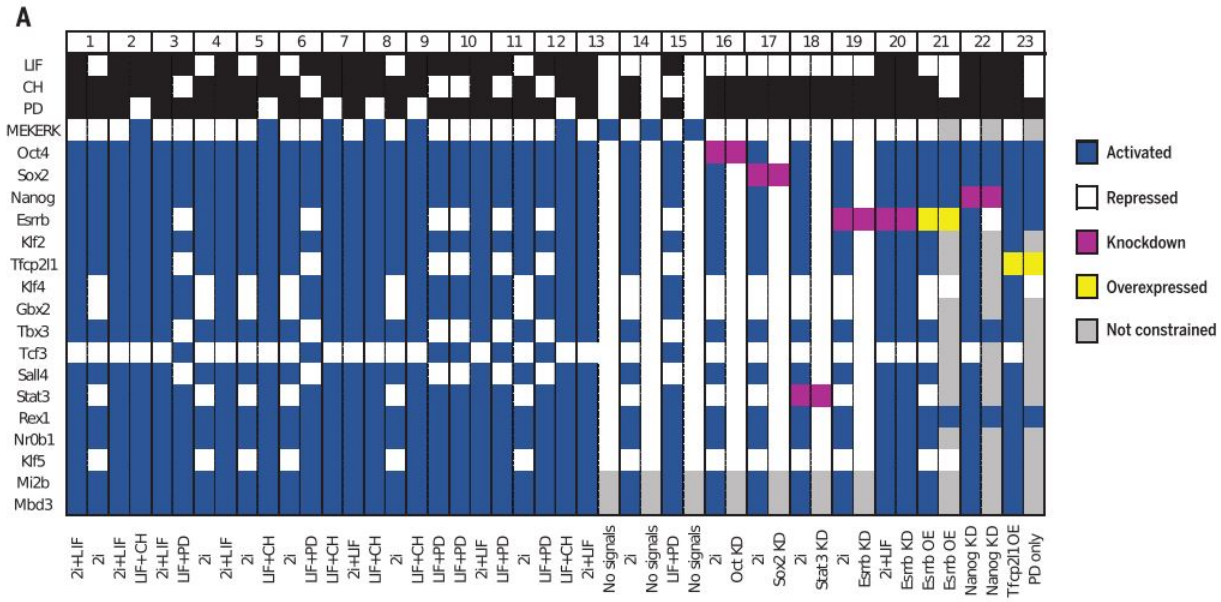

Figure 4: Experimental knockdown and culture perturbation results from Dunn et al. [31]. Every column pair (1-23) contains initial (left column) and final (right column) conditions either after knockdown, overexpression or a change of culture.

3 Project Overview

This project is constituted by two separate approaches to model induced pluripotency: a Boolean networks approach and a dynamical systems approach. Both approaches focus on modelling the transition from MEF cells to iPS cells that occurs when cells are being reprogrammed.

Central for both parts is the underlying genetic regulatory network topology of key genes in the reprogramming process. This topology, which will be used as a basis for both approaches, has been extracted from the ESCAPE database[1]. The network topology is shown in Figure 5.

3.1 Part One: Modelling the Reprogramming Process With a Boolean Network

The goal of this part is to model the full reprogramming process of MEF cells into iPS cells with a BN and use the model to predict experimental outcomes. We construct the

[1]ESCAPE [5] is an open source data base containing experimental data from analysis of human and mouse embryonic stem cells. It contains data from many recent diverse high-throughput studies including chromatin immunoprecipitation followed by deep sequencing, genome-wide inhibitory RNA screens, gene expression microarrays or RNA-seq after knockdown (KD) or overexpression of critical factors, immunoprecipitation followed by mass spectrometry proteomics and phosphoproteomics studies. From these it is possible to infer interactions between genes such as which genes are acticating/repressing which.

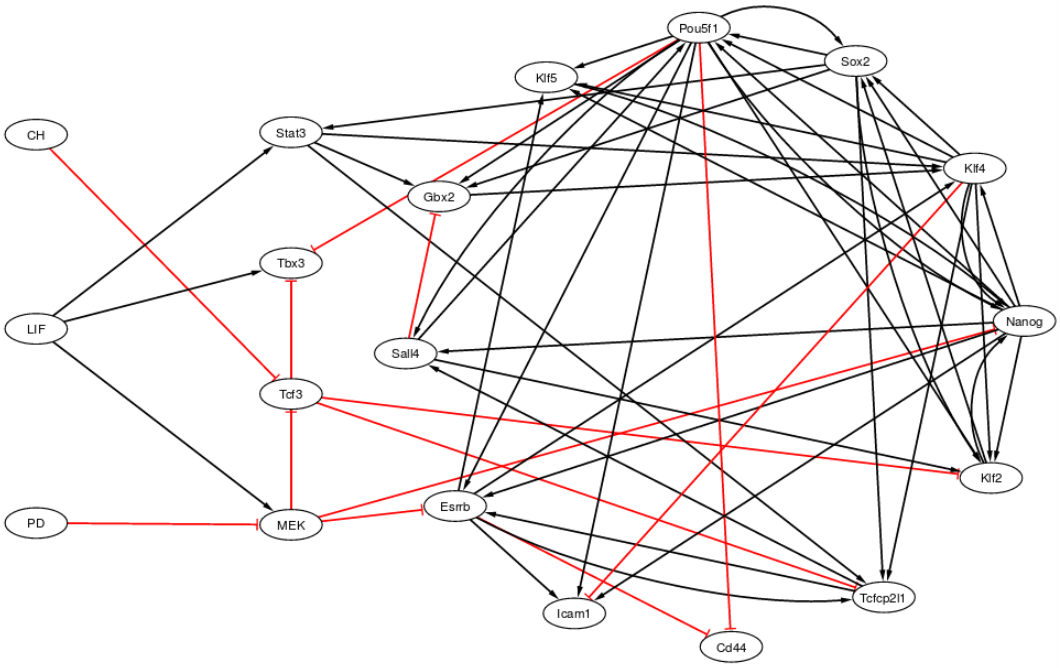

Figure 5: The topology of the gene regulatory network governing induced pluripotency. The genes and their interactions have been extracted from the ESCAPE database. A black directed arrow represents an activating interaction and a red blunted arrow represents repression. The nodes CH, LIF and PD are external culture nodes and thus have no incoming connections to them.

BN model by training the network topology in Figure 5 with the data from O'Malley et al. and Dunn et al. covered in Section 2.5. The BN model is additionally trained to replicate observed reprogramming efficiencies in different culture conditions. Dynamical simulations of gene KDs were then carried out and validated with additional experimental data provided by our collaborators at Kaji lab (group of Professor Keisuke Kaji). The result was a final BN that could accurately model the full reprogramming process and was capable of predicting experimental outcomes.

3.2 Part Two: Extracting a Minimal Dynamical Systems Network to Model Reprogramming

The motivation for this part of the project stems from the necessity of having simple models to describe intricate systems. Simple models can eliminate the need for extensive computations and furthermore increase our understanding of which the important regulatory players are. Therefore, we wanted to create a simplified dynamical systems model to simulate the reprogramming process and use it to find crucial interactions and gain new insights.

3.3 Thesis Outline

Immediately following this overview, we will present the methods behind creating and simulating the networks of both parts in this project. The methods section will be followed by the simulation results together with discussions. The thesis ends with a summary in the Conclusions and Outlook section.

4 Methods

4.1 Part One: The Boolean Network Approach

A Boolean network was applied on the genetic regulatory topology of Figure 5 and then used to simulate the transition from MEF cells to iPS cells. The results are presented in the next section. The following subsections describe the details of how the BN was created and how it was used to simulate induced pluripotency.

4.1.1 Creating and Training the Boolean network

Experimental results from the studies covered in section 2.5.1 and 2.5.2 were used as training data in the construction of the BN model. The training process is employed in order to fill in appropriate output entries in the truth tables of the nodes in the network. This is done such that the aforementioned results can be accurately reproduced when simulating the reprogramming process. The full training process, which involves iteration of several different intricate steps, is presented in Appendix A. The steps can be summarized as follows:

- Enforce attractors by filling in appropriate truth table outputs for each node
- Determine residual outputs by evaluating fitness of logical operators
- Compute network attractors and basins and train with respect to reprogramming efficiency (see Section 4.1.2 below)

In the last point above the network is trained to replicate the difference in reprogramming efficiencies observed in different culture conditions (i.e. different combinations of LIF, CH and PD). This requires a method for computing the efficiency of reprogramming *in silico* which will be introduced in the section below.

4.1.2 Quantifying Reprogramming Efficiency

This section will introduce a measurement of reprogramming efficiency *in silico* similar to the experimental Nanog-GFP CFP. As stated in Section 2.4, Nanog-GFP CFP can be used as a measure of how pluripotent cells in a culture are and can thus also be used as an indicator of cell reprogramming efficiency. If cells are quick to gain pluripotency, the efficiency of reprogramming is high and vice versa.

Previous researchers have used for instance specific attractor basins, such as the iPS-state basin, for similar tasks [19], which completely disregards the structure and behavior of the overall state space. Our method aims to bypass this problem by considering the full state space of the BN.

To accurately evaluate the global behavior of the BN, we devised a general distance measure from the iPS state, which will be called iPS proximity, which incorporates the basins of all attractors and their respective distances to the iPS state. The iPS proximity P for a BN with n nodes and M attractors with basins b_m and Hamming distances[1] d_m from the iPS state was defined as

$$P = \sum_{m=1}^{M} 5(20 - d_m)\frac{b_m}{3^n} \tag{4.9}$$

The iPS-proximity can be interpreted as an average basin-weighted distance for the attractors in the BN to the iPS state attractor. The factor $1/3^n$ normalizes the value of the iPS-proximity over the total number of states in a tree-level discretized system with n nodes (see Appendix A regarding the choice of a three-leveled discretization). The equation ensures that a BN will have an iPS proximity of 100 if the iPS state attracts all of state space and an iPS proximity of zero if the MEF state attracts all of state space (because the Hamming distance between the iPS state and MEF state was 20 in the model that was developend in this project). In a BN in which the MEF state is the only attractor the variables will hence take on the values $M = 1$, $d_1 = 20$ and $b_1 = 3^n$.

The iPS-proximity was hence used to quantify the simulated reprogramming efficiency and compared with experimental values for Nanog-GFP CFP. It will also be used in the validation process of the network covered in the next section.

4.1.3 Validating the Boolean Network

Validation of the BN was carried out in order to establish if the BN could reliably simulate the reprogramming process. This was done by simulating KD effect on the reprogramming efficiency of the BN in different cultures, then validating with similar experiments *in vitro*. The validation data set was provided by our experimental collaborators at Kaji lab, who conducted cell reprogramming experiments with KD of the relevant genes in the network.

4.1.4 Stochastic Analysis of the Boolean Network

By introducing a gene perturbation rate p, stochastic simulations was carried out and steady state distributions were obtained by means of Markov chain (MC) analysis. Establishing stochastic behavior is important when modelling real biological systems. The steady state distribution is a more 'real' estimate of the behavior of the BN because it takes

[1]The Hamming distance between two integer arrays of the same length is the sum of the pair-wise absolute differences of their entries.

into account random fluctuations and transition probabilities, which are always present in real biological systems.

Markov chains were created by continuously simulating stochastic updates of the BN until they reach the steady state distribution as indicated by the convergence test covered in Appendix B. The iPS proximity of the distributions were then calculated with Equation 4.9, but with the average distance of the total MC to the iPS state instead of the distances from the individual basins.

The important detail in the stochastic analysis is the choice of the gene perturbation rate p which was chosen to be $p = 0.05$. A summary of the rationale behind this choice is presented in Appendix C.

4.2 Part Two: The Dynamical Systems Approach

In order to extract a minimal network from the topology in Figure 5 we employed a bottom-up approach; we started from the simplest imaginable network and successively added modifications to see if it would be enough to encompass the interesting dynamics of reprogramming. Each iteration involved the following four steps:

- Merging different combination of nodes of the topology together to make a simplified network
- Setting up the dynamical equations of the network
- Determining model parameters via optimization
- Simulating deterministic and stochastic behavior

The procedures in each individual step are covered in detail in Appendix D.

If the behavior of the network then matches the key features of reprogramming, it may be a sufficient model to describe and predict reprogramming characteristics. The key features we wanted to address were the stability of the two states corresponding to MEF and iPS cells, and the regulation patterns of certain pluripotency-related genes.

The classification of genes with respect to their expression patterns was covered briefly in Section 2.5.1. To recapitulate, the three main patterns of interest for pluripotency-genes in the model were:

1. Gradual up-regulation of genes related to the iPS state, called late genes.
2. Rapid up-regulation of genes related to the iPS state, called early genes.
3. Transient down-regulation of other genes.

Networks that could accurately reproduce these behaviors were deemed potential candidates to model the generation of iPS cells.

5 Results and Discussion

5.1 Training Results for the Boolean Network

Given the gene regulatory network topology in Figure 5, the regulatory logic was learned following the procedures covered in the previous section with the help of the gene expression data presented in Section 2.5. The result was an ensemble of thousands of valid BNs with different regulatory functions. The training process was therefore extended to train the BNs with respect to experimental reprogramming efficiencies in terms of simulated iPS-proximity.

The experimental data of reprogramming efficiency in different culture conditions is shown to the left in Figure 6. The same figure also shows the iPS-proximity of the best performing BN. Recall that different culture conditions correspond to different combinations of CH, LIF and PD (input nodes in Figure 5), of which the culture with CH and PD together is called 2i. As evident from the figure, the BN could reproduce the effects of the real dynamics in the way that LIF is the most efficient reprogramming culture and 2i is the least efficient. The combination of them, 2i+LIF, is somewhere in between. The only difference is that the differences are greater in the experimental results.

The reasons for the small differences in the simulated case could be due to the few inbound connections from the external culture nodes CH, LIF and PD in the BN. There are only a total number of five interactions among them, one of which is only regulating the 'leaf'-gene[1] Tbx3. For the 2i+LIF and 2i cultures there are only two connections that differ (see Figure 5). This suggests that there could be important connections from CH, LIF or PD missing in our topology.

A majority of the BNs did in fact not capture the observed differences between the cultures at all, indicating again that the impact of CH, LIF and PD may be underestimated in our model. The network results presented in Figure 6 is the greatest similarity from among dozens of different trained BNs. Since the observed culture differences are so large, one should expect at least some of the trained BNs to reflect this, which was not the case. Another possible explanation for this is that the genes downstream CH/LIF/PD, such as Stat3 or MEK, ought to have more influence over the rest of the network. The influence of individual genes in the BN were not considered in this project, but is something that could be of interest in future research.

[1]Leaf-genes are genes which do not impact the other genes in the network (i.e. they do not have any outgoing connections).

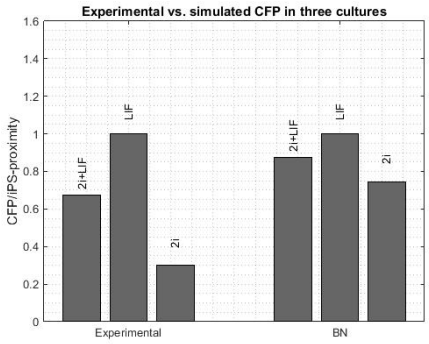

Figure 6: Training results showing reprogramming efficiencies of three different cultures (2i+LIF, LIF and 2i) in the experimental case (left) and simulated case (right). The experimental values are in terms of the normalized Nanog-GFP colony forming potential and the simulated values for the Boolean network are in terms of the iPS-proximity (Equation 4.9).

5.2 Validation Results for the Boolean Network

The BN above which was confirmed to reproduce the experimental colony efficiencies was selected for validation. This was carried out by predicting knockdown (KD) behavior of the BN and letting experimental collaborators perform the same studies experimentally. The predicted KD behaviors of the BN are displayed in Figure 7 (upper panel) together with the experimental results (lower panel).

The resemblances between the two cases are striking: almost all of the gene KDs have the predicted effect. All predictions are accurate in the 2i+LIF culture, and a total of only four KDs are predicted incorrectly in the LIF and 2i cultures put together, resulting in an overall accuracy of roughly 83%.

A closer look at the 2i+LIF culture, which was the one most similar to the experiments, is shown in Figure 8. There are no qualitative differences in this culture. The only difference between the predicted and experimental results comes from the level of which Nanog impacts reprogramming: the effect of Nanog KD seems to be more severe in the experimental case.

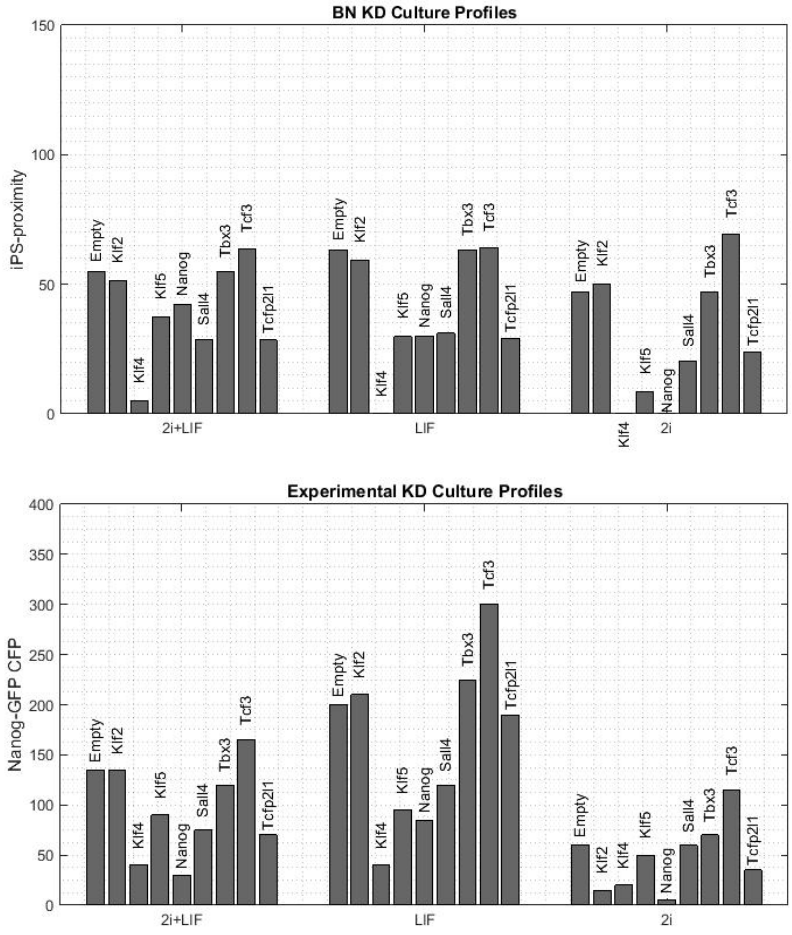

Figure 7: Computed reprogramming efficiencies of knockdowns in the Boolean network (BN, top) compared with experimental knockdown reprogramming efficiencies (bottom). The BN efficienciess are in terms of the iPS-proximity (Equation 4.9) and experimental values are in terms of the Nanog-GFP colony forming potential. Bar-labels indicate knocked down genes. With an overall prediction accuracy of 83%, the results show clear similarities between the BN predictions and experimental results, indicating that the BN model is in fact representative of the real gene-regulatory circuitry.

Many important remarks can be made regarding the results of the BN model in relation to previous studies:

- The severe effects of Klf4 KD are to be expected since expression of Klf4 is required for both self-renewal and pluripotency in embryonic stem cells (unless a substitute is introduced e.g. forced expression of Tfcp2l1) [33, 34, 35].
- While Klf2-KD stem cells have the ability to survive in LIF-conditions, they are also known to perish in 2i [36]. This is in line with both simulation and experiments (with the exception of simulated Klf2 KD in 2i culture).

- Nanog is considered a core element of the pluripotent transcriptional network and Nanog KD has been shown to induce differentiation in embryonic stem cells [33]. It is thus reasonable to expect Nanog KD to greatly impair reprogramming efficiency, which was indeed the observation in our results with the sole exception of the simulated 2i+LIF culture. In this culture, the effect was only moderate.

- Embryonic stem cells cultivated in 2i+LIF have been shown to tolerate singular loss of key TFs such as Nanog, which is consistent with our simulations (albeit not evident from the experimental results) [35, 37].

- The unaffected reprogramming efficiency of Tbx3 KD demonstrates that both simulations and experiments reflect the fact that Tbx3 is not necessarily required as a regulator in a model for pluripotency [31].

- The severe effects of KD in the 2i culture compared to 2i+LIF agrees with the observation that 2i+LIF is more robust to genetic perturbations than the 2i culture [31].

- Increased Nanog and Oct4/Pou5f1 expression has been reported as a consequence of Tcf3 KD, consistent with both simulation and experiments since Nanog and Oct4 are key reprogramming factors and should thus increase reprogramming efficiency [38].

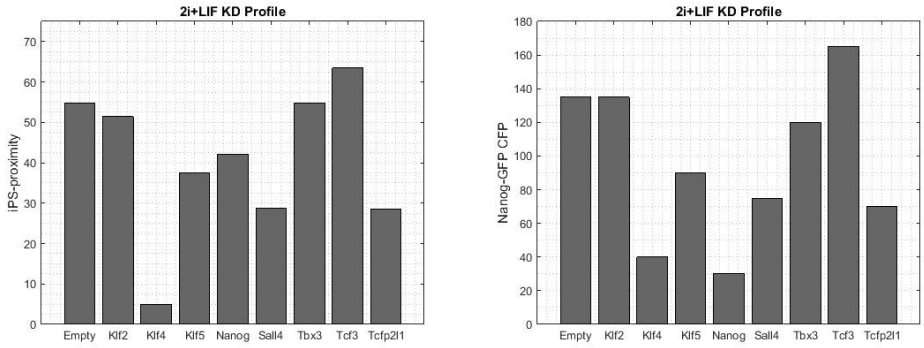

Figure 8: Knockdown predictions of the Boolean network (left) compared with the experimental results (right) in the 2i+LIF culture. Bar-labels indicate knocked down genes. The predictions are very similar to experiments, with the only difference between the two being the quantitative effects of Nanog knockdown.

Furthermore, by construction, the BN is consistent with the fact that Oct4/Pou5fl and Sox2 are indispensable when it comes to maintaining pluripotency since this behavior was already applied in the training process.

Note that the inter-culture differences are in general greater in the experimental case as was also observed in the non-KD training results in figure 6, indicating that the impacts of the input nodes (CH, LIF and PD) are still lower than intended. Modifying the influence

of individual nodes on-demand can be challenging from a computational standpoint, but is certainly something that can be investigated in future research. It is also possible that there are other valid BNs which mimic this behavior more accurately since the full set of possible BNs was not evalued exhaustively (see Appendix A).

The only significant disagreements between the predicted and experimental results are the predictions of Tcf3 & Tcfp2l1 in LIF, and Klf2 & Klf5 in 2i. The high agreement between simulation and experiments indicate that the BN is in fact representative of the real gene-regulatory circuitry, and can reliably be used to predict experimental outcomes.

Furthermore, the predictive results presented above validates the usage of iPS-proximity as a means of gauging reprogramming efficiency and the pluripotency of cells *in silico*. It is likely that the faulty predictions are due to factors such as defective regulatory logic or missing connections rater than inherent flaws of the iPS-proximity.

5.3 Results of Stochastic Simulations of the Boolean Network

To validate the BN in terms of its stochastic behavior, the steady-state distribution of the network was computed. The reprogramming efficiencies of the deterministic model are not enough to fully gauge the BN validity because real-life dynamics exhibit stochasticity. The stochastic reprogramming efficiencies are displayed and compared with the experimental results in Figure 9. The agreement between the stochastic and deterministic predictions are extremely high – only two predictions differ (Klf5 and Tcfp2l1 in 2i). Consequently, the remarks made in the previous section are applicable to the stochastic case as well. Importantly, this solidifies our BN as a reasonable model of cell reprogramming.

The only inaccurate predictions are the impact of Tcf3 and Tcfp2l1 in LIF and Klf2 and Tcfp2l1 in 2i. Since Klf2 in 2i was incorrectly predicted in the deterministic case as well, it is likely that the regulatory logic behind Klf2 is flawed.

Note that the differences in between cultures are now greater, making the stochastic results even more similar to the experimental results in this aspect, which alleviates perhaps the greatest drawback of the deterministic simulations.

It is worth noting that the steady state distribution is heavily dependent on the gene perturbation rate p. For a high enough p, the distributions will look exactly the same. On the other hand, a biologically sensible p is typically very small to reflect the rarity of gene perturbation. The simulations above were carried out for a gene expression rate of $p = 0.05$ (this choice is covered in Appendix C).

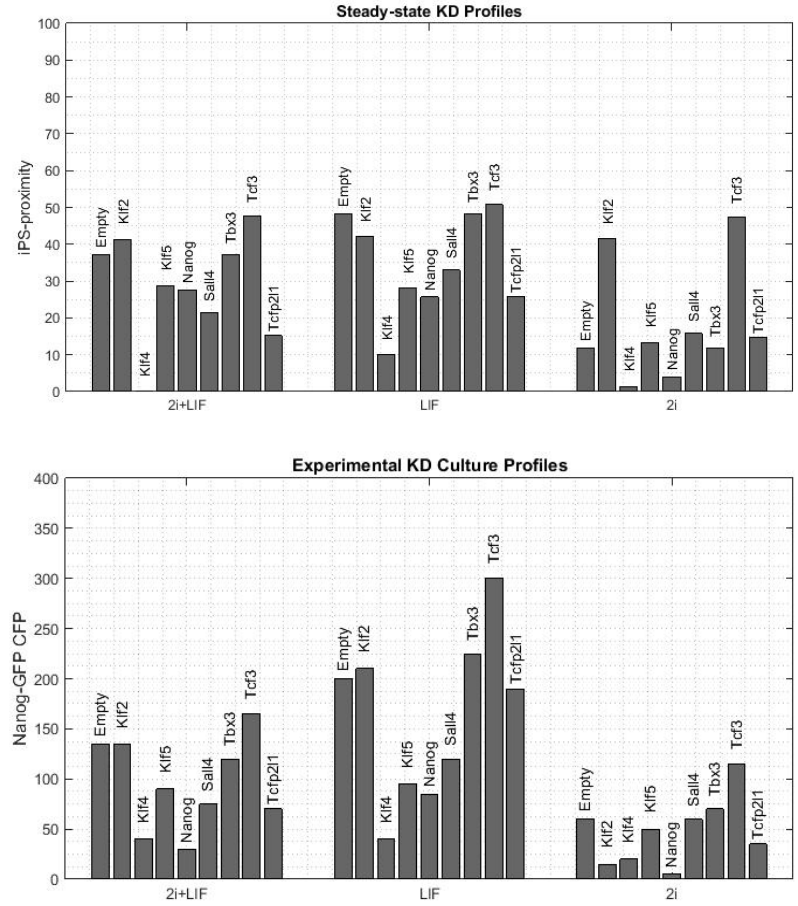

Figure 9: Simulated stochastic steady-state knockdown profiles of the Boolean network (top) in terms of iPS-proximity (eq. 4.9) together with the experimental knockdown results (bottom) in different cultures. Bar-labels indicate knocked down genes. Bar-labels indicate knocked down genes. The predictions show great agreement with the experimental results, indicating that the BN model also reliably simulates stochastic behaviors.

Overall, the results covered here suggests that the iPS proximity presented in Equation 4.9 is a very good indicator of the reprogramming efficiency of a BN even in the stochastic case. The only apparent drawback of it seems to be that it requires computation of the basins for all attractors in the deterministic case, which is not always practical for very large state-spaces (e.g. very large networks or more than three discretization levels).

5.4 The Minimal Dynamical Systems Model

The minimal dynamical systems model that still could encompass the full process of reprogramming is presented in Figure 10. This network managed to produce the three key behaviors of the reprogramming process: two stable states corresponding to MEF cells and iPS cells, early or late up-regulation of specific pluripotency-related genes, and gradual

down-regulation of other genes. The model does not distinguish between individual genes with similar expression patterns and instead models them together as ensemble-nodes.

The gene-expressions of the nodes in the network throughout the reprogramming process is shown and compared with experimental results in Figure 11. It is clear from the figure that the model accurately describes the time-evolution of pluripotency-related genes in reprogramming cells.

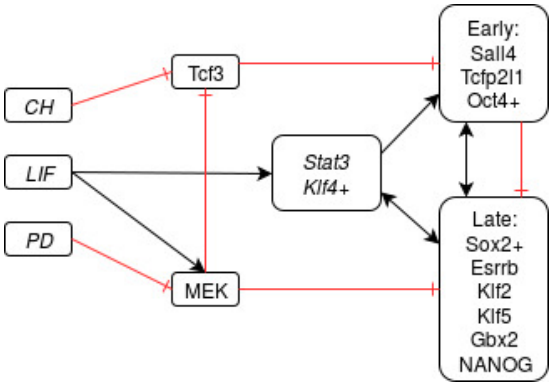

Figure 10: The proposed model for a minimal dynamical systems network that is still able to reproduce key aspects of the reprogramming process. Genes with similar characteristics are bundled together and considered as single pluripotency-nodes. Genes labelled with a plus sign are the reprogramming factors which are overexpressed in order to promote reprogramming.

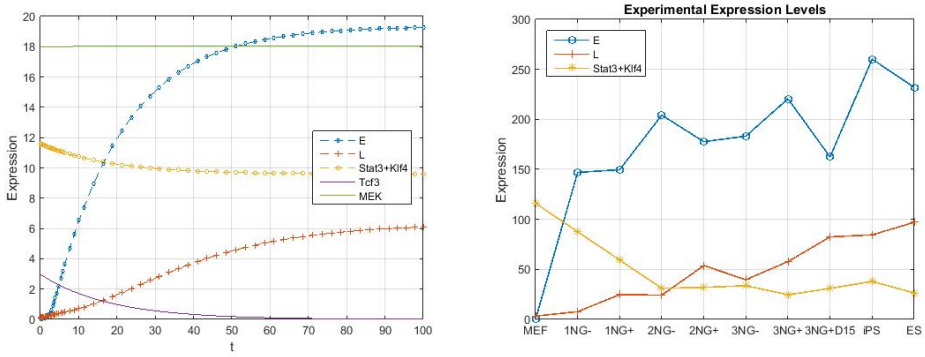

Figure 11: Simulated (left) and experimental (right) expression levels of genes in the reprogramming process. Note that the experimental results are for stages in the reprogramming process rather than a real time-series. The results show how the simple network in Figure 10 is able to produce the experimentally observed expression levels.

Figure 12 demonstrates the network's bistable nature in both the stochastic and the deterministic case. In a reasonable model for reprogramming, both MEF cells and iPS cells must exhibit stable dynamics both stochastically and deterministically. As seen in

the figure, this is indeed how the network behaves, which indicates that this network is an adequate model of the transcriptional network.

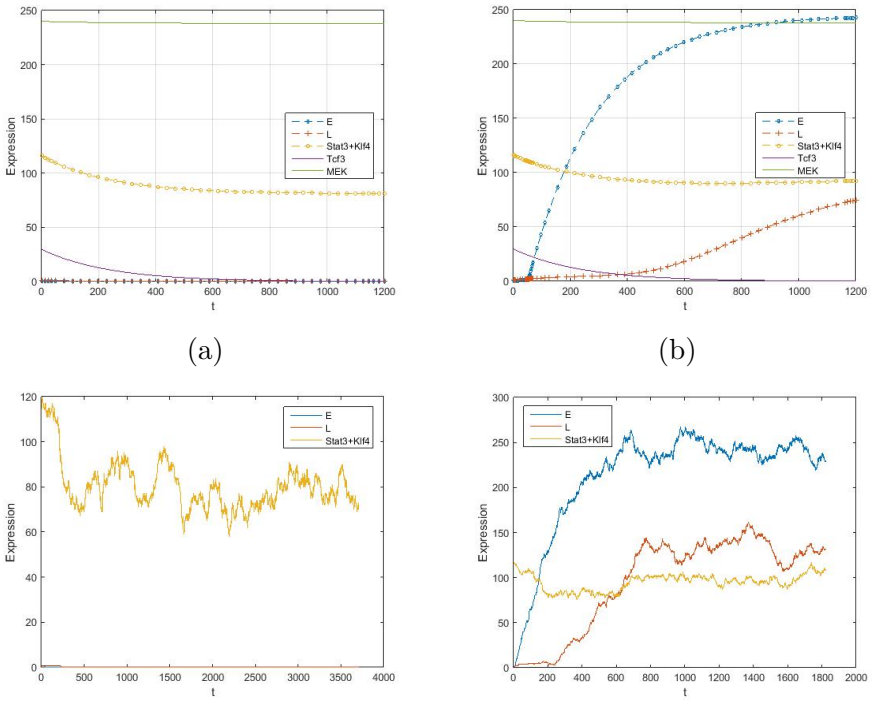

Figure 12: Stability of states in the dynamical system. (a) A deterministically stable low-expression state corresponding to MEF cells. (b) After over-expression is added, the MEF cells reprogram into high-expression iPS cells by increasing the expression of the early and late pluripotency-related factors (see Figure 10). (c) A stochastically stable low-expression MEF state demonstrating that cells do not reprogram spontaneously. (d) Stochastic stability of the high-expression iPS state demonstrated after introducing overexpression.

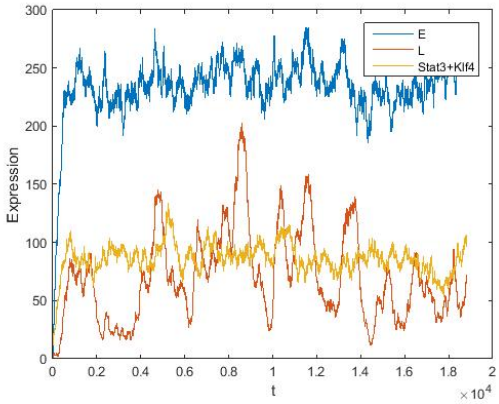

Figure 13: Long term stochastic behavior of the network in Figure 10. The results show very large long term fluctuations of the late pluripotency genes, suggesting that late genes may be reprogramming roadblocks.

In order to make reliable stochastic simulations like the ones in Figure 12 above, appropriate values for the initial conditions has to be determined. This is because the stochastic algorithm simulates individual molecular reactions such that the scale is no longer arbitrary (the available experimental data was normalized in relation to a specific housekeeping gene). The details and results of this scaling process is covered in Appendix E.

Long term stochastic simulations of the behavior of the network is presented in Figure 13. Interestingly, the results show very large long term fluctuations of the late pluripotency genes, suggesting that late genes may be reprogramming roadblocks. One can interpret the pluripotency of the network as being required to have late-gene expression levels above a certain threshold. This threshold then constitutes a barrier which the system has to stochastically overcome, thus being a reprogramming roadblock. A follow-up on these results where individual genes were separated from the late node in order to identify individual road-blockers is presented in Appendix F.

By comparing the previous experimental results shown in the lower panels of Figures 7 & 9 with the genes included in the late node, we may find indications of specific genes acting as roadblocks. Notably, KD of the late gene Nanog severely impacts reprogramming efficiency, suggesting that Nanog may be a major reprogramming barrier. A similar conclusion can be drawn for the Klf5 gene, which also impacts reprogramming, although to a lesser extent, when knocked down.

All of the above results indicate that the reprogramming process may be adequately modelled as a simple minimal model rather than a vast network of connections and interactions.

5.5 More Training Data May Severely Improve Boolean Network Performance

One issue with the training process of the BN was the lack of detailed gene expression data for the reprogramming process in different cultures. The studies covered in Section 2.5 provided invaluable training data for the network, but as with any training process, the more data the better. As mentioned in the section covering the construction of the BN (Appendix A.2), only about 99% of the truth table outputs had been determined after the initial training process. More training data would improve this ratio and perhaps even eliminate the need to add random attractors throughout state space (see Appendix A.4).

5.6 Threshold Boolean Functions and Other Functions Such as XOR are Disregarded

When determining the regulatory logic functions for genes in the network, only permutations and compositions of the functions AND, OR and NOT were considered (see Appendix A.2). This means that other Boolean functions which may be just as important and biologically feasible are not considered at all. For example, the exclusive OR (XOR) Boolean function likely exist within mammalian gene regulatory networks. In addition, Boolean functions which require several but not specific inputs to activate, so called threshold Boolean functions, are also missed. Such functions have been proposed to specifically exist within the pluripotency circuitry and could perhaps enhance simulation accuracy [43].

5.7 Knockdown of Overexpressed Klf4

One thing that may seem peculiar is the simultaneous KD and overexpression of Klf4 in the experimental validation studies in the results of Section 5.2. This can be done experimentally because the exogenous Klf4 is separate from the endogenous Klf4. The endogenous Klf4 is then knocked down as usual, but in order to not have the exogenous Klf4 interfere with the endogenous KD, the exogenous Klf4 is then removed. The effect of this is that Klf4 KD severely impairs reprogramming efficiency, since Klf4 is one of the overexpressed factors promoting reprogramming. This is done in the same way in the Boolean approach, resulting in the same effect. This should not be too surprising because a stable iPS state was never enforced in this condition (since there were no data points having Klf4 KD with which the BN could be trained; see Appendix A).

6 Conclusions and Outlook

In this thesis we have modelled the generation of induced pluripotent stem (iPS) cells from mouse embryonic fibroblast (MEF) cells which takes place when exposing MEF cells to reprogramming factors. We have constructed a Boolean network (BN) by an intricate training scheme and used it to predict experimental knockdown outcomes to an overall

accuracy of 83%. This suggests that the BN can be used for predictions and identifying genes which act as roadblocks in the reprogramming process.

We have suggested a quantitative measure for pluripotency and reprogramming efficiency of BNs and shown using experimental validation data that this quantity is indeed representative of the efficiencies of *in vitro* reprogramming in different culture conditions.

Additionally, we demonstrated by modelling the reprogramming process with dynamical systems that the process can be modelled by a simple network of interactions rather than a vast circuitry of connections. We have used this model to predict that certain genes with a specific type of expression pattern, called 'late' genes, are more likely to be roadblocking genes. These genes include Esrrb, Gbx2, Klf2, Klf5, Nanog and Sox2. According to our results, particularily Nanog (along with Klf4), and to a lesser extent Klf5, seems to be the major factors.

A next task from here could be to further improve the BN by enforcing behaviors that were not in line with validation data. One way to investigate this could be to look at quantitative measurements of the BN structure such as the influence or sensitivity of specific genes. Another possibility is to look at how the BN performs in conjunction with other BNs in the form of a PBN. Such studies could assist in eventually defining a complete description of the reprogramming process.

A The Boolean Network Training Procedure

Several toolboxes and software packages have been developed for application of BNs [39, 40, 41], but due to the specific control requisites in this project we opted to not use these predefined packages and instead developed an independent system. For instance, we wanted to be able to make simulations with trinary logic, where each node can take one of three values (0, 1, or 2) instead of the standard binary logic. This choice is based on the following two reasons:

- *Decreased similarity between prescribed attractor states* – with only two levels of logic, some experimentally observed states becomes virtually indistinguishable from one another because of reduced variability in the gene expression levels. Increasing the resolution prevents this.
- Observed barriers in the reprogramming process (called roadblocks) can possibly be due to some genes not reaching their maximum expression levels, thus being stuck in an intermediate state (i.e. with an expression level of 1 compared to a maximum of 2). We wanted to be able to capture this kind of behavior which cannot be done in binary logic.

The way overexpression of genes was modelled in the BN was by implementing additional overexpression-nodes. These nodes inherited their interactions from their endogenous counterparts, but cannot be reduced below a value of two when overexpression is active.

Overall, training and validation of the BN is a fairly long process in which some steps may not seem very straight forward. To help the reader keep track, a flowchart of the process is presented in Figure 14. The flowchart illustrates the whole process from initializing the BN topology to producing quantitative results. The individual steps are gone through in detail below.

A.1 Implementing a Topology, Prescribing Attractors and Creating Truth Tables

The first part of the training process is filling in the truth tables (TTs) of the nodes in the BN. For this, training data sets are needed, which were provided by the previously mentioned studies of O'Malley et al. [32] and Dunn et al. [31] covered in Section 2.5.

BN attractors were created out of the observed reprogramming stages in the O'Malley study. It is likely that the stages in their experiments should correspond to attractors in the network because the cells typically stay in each state for a while before transitioning to other stages. Although the average time the cells tend to stay in one stage varies between stages, the general trend is that they do not transition directly from MEF to iPS. The stages can be applied directly to the BN as attractors because the O'Malley study includes exact gene expression values for each of the stages. Their data was also already discretized/clustered (by K-means clustering) into the three different expression levels.

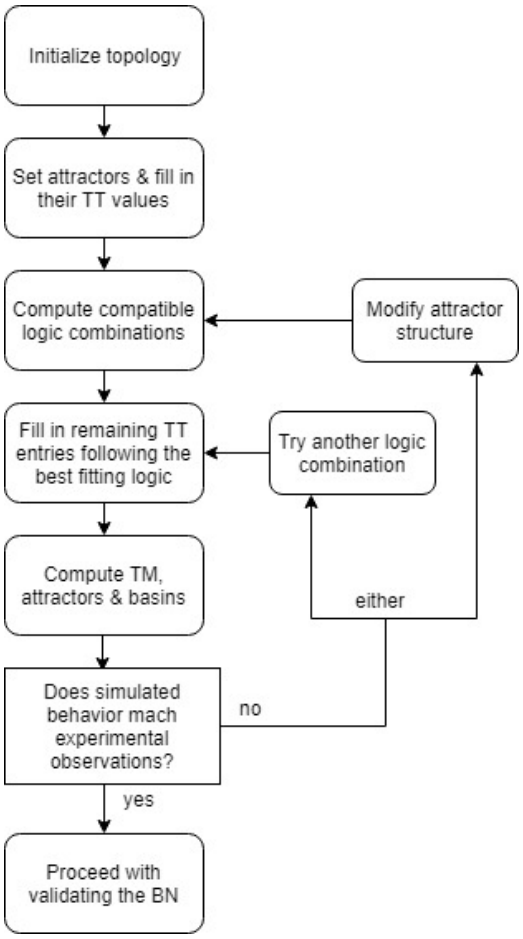

Figure 14: Flowchart of the BN training procedure. TT=truth table, TM=transition matrix (used to calculate attractors and basins). The iPS-proximity is calculated via Equation 4.9. All the steps are described in detail in this appendix. After training is finalized, the BN is validated with respect to new validation data sets.

The way the attractors are implemented is by filling in the appropriate TT output for each gene. As an example, consider a three-node network where we want the attractor state (011). Let each node have both other nodes as input. All three nodes then have TTs in the form of Table 2. To apply an attractor state of (011), fill in the corresponding output values for each node, i.e. $f_1(1,1) = 0$, $f_2(0,1) = 1$ and $f_3(0,1) = 1$.

It is important to note that there is no guarantee that all attractor states are compatible with each other. If, for instance, we also wanted to make an attractor state out of (111) in the above example, the attractors would clash. This is because that would require $f_1(1,1)$ to be equal to 0 as well as 1, which is not possible.

From the O'Malley paper, there were six different intermediate attractors (labelled 1NG-, 1NG+, 2NG-, 2NG+, 3NG- and 3NG+) apart from the predominant MEF-state and iPS-state attractors. All of these were implemented in each of the three different

Table 2: Truth table layout for a node with two inputs.

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

cultures. However, all of these attractors were not compatible with each other; for instance the NG+ and NG- variants of each attractor were particularly close to each other, with only two gene expression level differences. This led us to discard some attractors until all of them could fit. In the end, only the 3NG- and 1NG- stages were compatible with the MEF and iPS states simultaneously.

A similar routine was used to implement the knockdown studies and culture perturbations from Dunn et al. [31]. Their publication contains initial and final conditions for changes of culture as well as KD of genes (see Figure 4). The final conditions in their results are states where the network has stabilized and should thus be attractors in the BN. However, the binarized data in the Dunn study is potentially problematic for the trinary logic employed in this project. The binary values (0,1) were therefore translated into (0,2) in the trinary case. If the data from Dunn would then clash with the attractors from O'Malley, the clashing values were changed to 1 instead. If a clash would still occur after that, the O'Malley study was given priority over Dunn since the data from the O'Malley study was obtained by studying the transition from MEF to iPS whereas the Dunn study is only for cells already in the iPS stage.

At this point we have created truth tables for each node such that the BN will inherit the attractor structures of the experimental training data. However, the truth tables are incomplete because not all the outputs are filled in. The next part of the training process address this issue.

A.2 Determining Logical Operators and Filling the Truth Tables

This section covers the process of filling in the remaining truth table outputs left from the previous section. The goal here is to make the nodes of the network behave in a biologically sensible way in cases which the training data from O'Malley and Dunn did not explicitly cover.

From the previous step in the training process there were about seven different attractors enforced in all three cultures. This means that only 336 truth table outputs were determined since there are 16 dependent genes in the network. Compared to the total number of TT outputs, this translates to roughly 0.9%, and another 99.1% of the TTs has yet to be filled in. The next step in the training process is filling in these missing values.

As mentioned in Section 3.1 Boolean operators such as AND, OR and NOT have direct

implications in a biological transcription process. There are of course other logic gates as well but in order to keep the model as simple as possible we wanted to minimize the number of logic gates to consider, and therefore only considered the gates AND, OR and NOT.

The idea here is to determine which logical operators, and which permutations and compositions of them, are in best agreement with the 336 TT outputs. The rest of the outputs are then filled in according to these functions. In the case of repression, the function was set to NOT by default (i.e. if A represses B then NOT(A) was set as the input to B rather than just A). Then, all the possible combinations and perturbations of the two remaining operators were tested exhaustively. The fitness s of each combination was calculated as the sum of absolute differences between the TT outputs and the logic function outputs, i.e.

$$s_f = \sum_k |TT_k - f(x_1^{(k)}, \ldots, x_{p_i}^{(k)})| \tag{A.10}$$

for logic function f and gene i with p_i parents where the sum goes over rows in the TT.

For most genes there were a handful of different functions with the same fitness value. In these scenarios, one of them were selected arbitrarily as the generator function for the rest of the TT. This means however that there are several different BNs which up to this point are equally valid.

The exogenous over-expression nodes were excluded from this training process. Instead the rules for them were set to follow their endogenous counterpart.

After this is done, all the genes have their TTs fully determined. From this point on, simulations of the behavior and dynamics of the BN is therefore possible.

A.3 Computing Transition Matrices, Attractors and Basins

The purpose of computing the transition matrices (TMs) is to later use them to calculate the attractors and their basins of attractions. As previously stated, the attractors and their basins are paramount in the study of GNRs because they are associated with specific phenotypes. This section covers the methods for computing the TMs and using them to calculate the attractors and their basins.

Computing the TM of the BN means filling in the entries of the 1×3^n array of state transitions, where entry i holds the next integer state $x \in \{1, \ldots, 3^n\}$ for the output state i. The first thing to note here is that only dependent genes are included when counting the dimension n of the network. This means that the culture-nodes CH, LIF and PD (see Figure 5), as well as the over-expression nodes of exogenous Oct4, Sox2 and Klf4, are not included because they are static. Furthermore, the nodes Tcf3, Cd44 and Icam1 actually have no outgoing connections and will hence not impact the next state of the BN (they are so called leave-nodes). The task of computing the TM is thus reduced to computing its 3^{13} entries instead of the original 3^{19}. This is done separately for the three different cultures.

The following two algorithms were implemented to compute all single state and cyclic attractors and their respective basins:

Input: Transition matrix TM.
Output: All single state and cyclic attractors.
Generate an array a of size 3^n for integer states $x \in \{0, \ldots, 3^n\}$ containing information whether or not state a_i has been visited before. Initialize all a_i's to zero;
for *each state* $i = 1$ **to** 3^n **do**
    if $a_i = 1$ **then**
        continue;
    else
        Look up successor states for i repeatedly and set their corresponding a's to 1 until a previously visited state is reached. If the same state is visited twice in a row, that state is a single-state attractor. If the visited state is one of the successor states of a_i, the visited state is the start of a cyclic attractor. If neither is true, the state is not an attractor.
    end
end

Algorithm 2: Finding Single-state and Cyclic Attractors.

Input: Transition matrix TM, state x of which the basin is computed.
Output: Basin of attraction B for state x.
Find all TM entries with state x as output and save them in an array a;
Initialize B to the number of states in a;
Initialize a counter *diff* which counts the number of new states in every generation;
while *diff* $\neq 0$ **do**
    Find all TM entries with states in a as an output. Save them in an array b;
    Set *diff* to length of a minus length of b;
    Set $B = B +$ *diff*;
    Set $a = b$;
end

Algorithm 3: Finding the basin of attraction of a single state. The algorithm computes all pre-images of a specific state with also includes ones that are part of a cyclical attractor, such that only one state of a cyclic attractor is needed to compute its basin.

There are several other ways of computing attractors and basins without using the TM, which could be useful because computing the TM can be very computationally tedious. Take for instance the general reverse algorithm; it computes the basin for a given state by effectively running the updates backwards (the algorithm is presented in Appendix G). This algorithm was implemented to our BN early in the process, but it turns out that this algorithm is extremely slow for states with very large basins. The algorithm is also incapable of finding unknown attractors which may be very dominant over the BNs behavior.

Following these procedures of computing the attractors and their basins, all of the relevant information of the structure of the BN is known. The thing that remains is evaluation of the BN's performance.

A.4 Evaluating the Boolean Network Behavior

Now that all the attractors and their corresponding basins are known we can evaluate the BN by looking at its structure or comparing its dynamics to real experimental observations. If the BN is not in line with the desired behaviors, modifications are made and the training process is started over.

Initial studies of the BN after computing its attractors and their basins revealed that it is incredibly attracted to the iPS state – the basin of the iPS state was composed of an astounding 99.9% of state space. Because this is not in line with the fact that reprogramming is a very cumbersome process, modifications were made to the BN in order to change the basins. This was successfully achieved by adding random attractors throughout the state space with the same procedures as in Appendix A.1.

Another issue with the BN was that it didn't reflect the differences that had been previously observed in reprogramming efficiency between cultures. To account for these, the goal was to make the basin of the iPS state larger in cultures which were better at reprogramming and vice versa. We discovered that the basins in the different cultures could be made dissimilar by slightly changing the expression level of specific genes between cultures. For instance, the MEK gene could be modified from zero in all three cultures to 0, 1 and 2 when the attractors are implemented (see Appendix A.1). We tried to limit the changes to genes which were directly impacted by the culture nodes, namely MEK, Tcf3 and Stat3. In the end, the following changes were decided upon:

- MEK expression level changed to 0/2/1 for all states in cultures 2i+LIF/LIF/2i
- Stat3 expression level changed to 2/2/0 for all states in cultures 2i+LIF/LIF/2i
- Tcf3 expression level changed to 1/2/0 for all states in cultures 2i+LIF/LIF/2i

The changes above follow from intuitive arguments for the expression levels of the genes in relation to their connections. For example, the MEK gene should not be on when PD is on and LIF is off because PD represses MEK and LIF activates it. This step successfully diversified the basins, but didn't manage to change them in the desired manner (2i, which is the least effective culture, had the largest iPS basin by far).

Finally, an activating relation of Nanog by Stat3 was implemented in the topology of the BN, which managed to produce the desired culture differences.

B The Kolmogorov-Smirnoff Test for Testing Convergence

To test whether the Monte-Carlo updates of the Boolean network had approached the steady state-distribution, the Kolmogorov-Smirnoff test (or KS-test for short) was used to test the convergence. The test checks whether two empirical distributions differ enough to conclude that they come from different probability distributions. The idea is that, if the Markov chain (MC) is stationary, two distributions $\pi^{(t_1)}$ and $\pi^{(t_2)}$ are the same for arbitrary times t_1 and t_2. The KS-statistic for a MC is defined as

$$D = \frac{1}{M} \max_s \Big| \sum_{m=1}^{M} \mathbb{1}_{\{x \leq s\}}(\boldsymbol{x}_1^{(mt)}) - \sum_{m=1}^{M} \mathbb{1}_{\{x \leq s\}}(\boldsymbol{x}_2^{(mt)}) \Big| \tag{B.11}$$

where the indicator function $\mathbb{1}_{\{x \leq s\}}(x)$ equals one if $x \leq s$ for lexicographically ordered states $x \in \{0, \ldots, 2^n\}$ and zero otherwise. In other words, $\sum_{m=1}^{M} \mathbb{1}_{\{x \leq s\}} \boldsymbol{x}_n^{(mt)}$ is the cumulative visits to states lower than s in the Markov chain $\boldsymbol{x}_n$ of length M. The KS-statistic can be used to build confidence bands of the underlying distribution. By defining

$$F_n = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}_{\{x \leq s\}}(\boldsymbol{x}_n^{mt}) \tag{B.12}$$

the Dvoretzky–Kiefer–Wolfowitz inequality gives bounds for the probability that the random function F_1 differs from the random function F_2 by an amount greater than ϵ for a given constant $\epsilon > 0$ like so:

$$P\Big(\sqrt{\frac{nm}{n+m}} \max_x |F_1(x) - F_2(x)| > \epsilon\Big) \leq 2e^{-2\epsilon^2}, \tag{B.13}$$

for two empirical distributions F_1 and F_2 of length n and m. If ϵ is chosen such that $\alpha = 2e^{-2\epsilon^2}$, the null hypothesis[1] is rejected at level α if $\sqrt{\frac{nm}{n+m}} \max_x |F_1(x) - F_2(x)| \geq \epsilon$. For instance, at a significance level of $\alpha = 0.05$ and distribution lengths $n = m = 10^3$ the largest difference between the distributions has to be greater than roughly 0.06 in order to reject the null hypothesis.

To get two quasi-independently and identically distributed samples for the test, one can select two samples $\mathbf{x}_1^{(t_1, t_1+\Delta m, \ldots, t_1+\Delta M)}$ and $\mathbf{x}_2^{(t_2, t_2+\Delta m, \ldots, t_2+\Delta M)}$ for $t_1 < t_2$ and $t_2 - t_1 \neq \Delta m,\ 0 < \forall m < M$ [10].

[1]The null hypothesis in this case is that F_1 and F_2 comes from the same underlying distribution function.

C The Gene Perturbation Rate in Stochastic Simulations

This appendix covers the choice of the gene perturbation rate p which is used when simulating stochastic BN behavior. The choice of $p = 0.05$ comes from comparing the perturbation matrix of a binary system with that of a trinary system (see Appendix A for details about the use of trinary logic). The transition probability matrix of a BN has been shown to be the sum of two separate matrices, of which only one is related to gene perturbation: the perturbation matrix [22].

The perturbation matrix of a binary system is defined as

$$A = p^{h(x_1,x_2)}(1-p)^{n-h(x_1,x_2)}\mathbb{1}_{x_1 \neq x_2} \tag{C.14}$$

for n genes and Hamming distances $h(x_1, x_2)$ between states x_1 and x_2, which translates to

$$A = p_2^{2h(x_1,x_2)}(1-p_2)^{2n-2h(x_1,x_2)} \tag{C.15}$$

in a trinary system. In the trinary case the distances from a high to a low expression value is two rather than one, and one can preserve the transition probability from a high to a low state by letting

$$p_1^{h(x_1,x_2)}(1-p_1)^{n-h(x_1,x_2)} \sim p_2^{2h(x_1,x_2)}(1-p_2)^{2n-2h(x_1,x_2)} \tag{C.16}$$

for the binary perturbation rate p_1 and the trinary perturbation rate p_2. This can be solved for $p_1 \approx 0.0025$ (a suitable level for a binary system according to [10]) resulting in a perturbation rate $p_2 \approx 0.05$ in the trinary system.

D The Dynamical Systems Approach: Detailed Procedures

This appendix covers the four steps of the selection process for finding a minimal dynamical systems network. Iteration of these steps for several different network topologies finally resulted in the simplified network that is covered in the results of secction 5.4.

D.1 Merging Nodes of the Topology Together to Make a Simplified Network

To create a simplified network, a bottom-up approach was employed where the simplest imaginable network was tried first and then gradually elaborated upon. First off, it is easy to see that a network with a single node can't possibly capture any interesting dynamics. However, with more nodes being added, the potential dynamics becomes exponentially more complex. For instance, [42] has shown that a simple genetic bistable switch can be constructed in *E. Coli* bacteria with just two dynamic nodes.

When merging nodes, the independent nodes are left out (CH, LIF and PD), and the other nodes are bundled together and modelled collectively. This means that the bundle-node inherits all the incoming and outgoing interactions from all of its component nodes to create a new topology. The genes Tcf3 and MEK were also separated from the merging process because they are not strictly related to pluripotency. As an example of how this can look, see Figure 15 below.

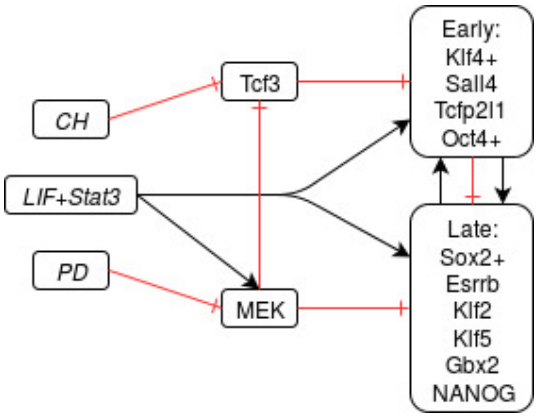

Figure 15: An example of a simplified topology of the GRN in Figure 5.

By trying out many different topologies and assessing their behaviors according to the steps that follow, a final topology could later be decided upon, which most accurately replicated experimental data. This final topology is the one presented in Figure 10 in the results of Section 5.4.

D.2 Setting up the Dynamical Equations of the Network

After obtaining a simplified topology, the network were modelled by following the Shea-Ackers formalism covered in Section 2.3.1. This entails setting up the system of ODEs in the form of Equation 2.8 which governs the dynamics of the network.

For example, the equations describing the final topology in Figure 10 are set up as follows:

$$\frac{d[E]}{dt} = \frac{\beta_1[E]^{\delta_1} + \beta_2[L]^{\delta_2} + \beta_3[S3K4]^{\delta_3}}{1 + \beta_1[E]^{\delta_1} + \beta_2[L]^{\delta_2} + \beta_3[S3K4]^{\delta_3} + \beta_4[Tcf3]^{\delta_4}} - \gamma[E] + \alpha_1 \tag{D.17}$$

$$\frac{d[L]}{dt} = \frac{\beta_1[E]^{\delta_1} + \beta_2[L]^{\delta_2} + \beta_3[S3K4]^{\delta_3}}{1 + \beta_1[E]^{\delta_1} + \beta_2[L]^{\delta_2} + \beta_3[S3K4]^{\delta_3} + \beta_5[MEK]^{\delta_5}} - \gamma[L] + \alpha_2 \tag{D.18}$$

$$\frac{d[S3K4]}{dt} = \frac{\beta_2[L]^{\delta_2} + \beta_3[S3K4]^{\delta_3} + LIF}{1 + \beta_1[E]^{\delta_1} + \beta_2[L]^{\delta_2} + \beta_3[S3K4]^{\delta_3} + LIF} - \gamma[S3K4] + \alpha_3 \tag{D.19}$$

$$\frac{d[Tcf3]}{dt} = \frac{\beta_4[Tcf3]^{\delta_4}}{1 + \beta_4[Tcf3]^{\delta_4} + CH} - \gamma[Tcf3] \tag{D.20}$$

$$\frac{d[MEK]}{dt} = \frac{\beta_5[MEK]^{\delta_5} + LIF}{1 + \beta_5[MEK]^{\delta_5} + LIF + PD} - \gamma[Tcf3] \tag{D.21}$$

D.3 Determining Model Parameters Via Optimization

Before solving the system of ODEs, suitable parameters in the dynamical equations must be set up. These include the binding affinities β, the Hill coefficients δ, the decay constants γ, and the over-expression constant α (if applicable) for every equation in the system. These are determined by optimizing the solutions to the analytically ODEs toward a target function with respect to some objective function. Time series for gene expressions throughout the reprogramming process data provided by the O'Malley study were used as the target function. In other words we seek the parameters which minimizes the difference between the analytically solutions and the experimental time series. A standard root mean square error was used as the objective function, defined by

$$RMSE(\mathbf{p}) = \sqrt{\frac{\sum_t^N ([C]_t^{(exp)} - [C(\mathbf{p})]_t^{(model)})^2}{N}}. \tag{D.22}$$

for a set of parameters $\mathbf{p}$ and an experimental series of measurements of length N.

To model reprogramming, the system is required to be at least bistable with one stable MEF-state and one stable iPS-state. This type of behavior is enforced by following these two steps during optimization:

1. Compute non-overexpressed stability concentrations. This state should correspond to the MEF state, then

2. If the expression levels of any of the iPS genes are higher than a designated 'MEF-threshold', set the objective function value to infinity.

The threshold value was chosen to be 20% of the lowest iPS-state target value. These two steps forces the system to posses a stable state with low pluripotency-gene expression levels, namely a MEF-like state. It doesn't directly impact the existence of an iPS-state, which is instead addressed by the high target values in the target function (i.e. the experimental time series).

The method above makes the objective function discontinuous which in turn makes gradient-based optimization algorithms such as global search and multistart solvers unusable. Furthermore, it creates plateaus of undesired high-objective-function values in the function landscape which makes simulated annealing approaches sub par to other non gradient based algorithms such as pattern searches or genetic algorithms. The chosen optimization algorithm was therefore a hybrid pattern-search/genetic algorithm, which searches the state space for a starting location with a genetic algorithm and then switches to pattern-search (see the MATLAB documentation on `patternsearch` for additional information).

When implementing these algorithms with MATLAB it is useful (although not required) to set parameter bounds for the algorithm to search within. The choices of the parameter bounds are discussed in the subsection below.

D.3.1 Optimization-Parameter Bounds

The following are the thought-processes behind selecting the parameter bounds for the optimization of the parameters in the dynamical systems approach:

Binding affinity (β): The lower bound for the concentration coefficients was set to zero, meaning that the specific promoter doesn't have to be bound in order for transcription to occur. The upper bound was set to 20 because values above this produced saturated concentrations far higher than the target values.

Hill coefficients (δ): A lower bound for the Hill coefficients was set to 1 which corresponds to a standard linear response. The upper bound was set to five because a value of five and higher is tantamount to just a step function.

External factors (CH, LIF, PD): The range for the external factors was set to [0, 100]. The lower bound means that the external factor is not present and the upper bound was set because at 100 the equation then effectively reaches the limit $\lim_{x\to\infty} \frac{x+...}{1+x+...}$.

Decay coefficient (γ): Gamma is perhaps the most important of the parameters because it controls the time resolution of the dynamic variables. It dictates the probability per unit time that a TF will decay. Within this project we assumed that this was constant and the same for all TFs. A suitable range for gamma was deemed to be [0.05, 0.2] due to the

fact that lower or higher values produced saturation concentrations very far off the target values.

Over-expression factor (α): When over-expression is present a value of $\alpha = 0$ makes no sense because it means that there is no overexpression. Alpha started to affect the dynamics at a value of about 0.01, which was therefore set to the lower bound. An alpha value over one made the concentrations saturate well over the target values and thus 1 was thus set as the upper alpha bound.

An additional detail in the optimization sense is the '`TFINAL`' parameter of the `ode45` MATLAB function which dictates the time up to which the function is integrating the system of ODEs. Of course, the system must have enough time to reach a steady state, but too much time skews away the interesting dynamics in the plots. This was counteracted by running until the standard deviations of the expression levels for the last ten steps were beneath a value of one.

D.4 Simulating Deterministic and Stochastic Behavior

The last step of the selection process is simulating the dynamics of the network and assess whether the topologies are biologically reasonable. Both stochastic and deterministic simulations were carried out in order to fully grasp the behavior. Deterministic results are obtained easily by solving the system of ODEs of concentrations, which in this project was done by applying the MATLAB function `ode45` which uses a fourth order Runge-Kutta solution method. It was assumed that the accuracy of this method was good enough because the most important results were qualitative rather than quantitative.

For an illustration of how a network behaves deterministically and how the stability of the MEF and iPS states can be confirmed at this stage, see Figure 16. The figure shows the stable MEF and iPS states of the network in Figure 15 and demonstrates how the theoretical background of the stability of states can be visualized in terms of the nullclines. It is also apparent from Figure 16(c) that the late and early genes do not exhibit their desired behavior, which led us to discard this particular network.

Stochastic behaviors of the dynamical systems were simulated by the Gillespie algorithm which is covered in the section below.

D.4.1 The Gillespie Algorithm for Stochastic Dynamics

The Gillespie algorithm is the most widespread stochastic algorithm for modeling enzyme reactions. Mathematically, it uses dynamic Monte Carlo steps to decide the next type of reaction to occur based on the different reaction rates. Thereof each reaction has a separate rate associated with it which is typically linked to the concentration of the respective promoting gene/enzyme/protein. This results in a continuous time Markov Chain (MC) with a probability density function described explicitly by

$$P(\tau, \mu) = \underbrace{a_0(\mathbf{X})e^{-a_0(\mathbf{X})\tau}}_{\substack{\textit{factor related to} \\ \textit{elapsed time}}} \cdot \underbrace{\frac{a_\mu(c_\mu, \mathbf{X})}{a_0(\mathbf{X})}}_{\substack{\textit{factor related to} \\ \textit{next reaction}}} \tag{D.23}$$

for a state space $\mathbf{X}$ with rate functions $a_\mu(c_\mu, \mathbf{X})$ and $a_0(\mathbf{X}) = \sum_\mu a_\mu(c_\mu, \mathbf{X})$, where c_μ is the concentration of reactant μ. Formally, this PDF describes the probability that the next reaction will be a μ-reaction and occur within a time interval $(t,\ t+\tau)$ given a state $\mathbf{X}$ at time t. The rate function is calculated for each reactant from Equation 2.7 with the important difference that the last decay term $\gamma[TF_i]$ is considered as a separate reaction and thus have a μ-index of its own.

The algorithm draws τ and μ from the PDF by generating uniformly distributed random numbers $r_1, r_2 \in [0, 1]$ and letting

$$\tau = \frac{1}{a_0} \ln \frac{1}{r_1} \tag{D.24}$$

and

$$\mu = \underset{\mu}{\operatorname{argmin}}\ (a_0 r_2 \leq \sum_{i=1}^{\mu} a_i). \tag{D.25}$$

The algorithms is presented below. The choice of t_{max} is naturally such that the system has enough time to reach a stable state.

Input: Initial concentrations $\mathbf{X}_0$, maximum time t_{max}.
Output: Time series for concentrations $\mathbf{X}$, array of timestamps $\mathbf{s}$.
Set $t = t_0$;
Compute a_μ and a_0 for the state $\mathbf{X}$;
while $t < t_{max}$ **do**
  Generate τ and μ using Equations D.24 and D.25;
  Update the time $t = t + \tau$ and add t to $\mathbf{s}$;
  Update concentrations $x_\mu = x_\mu \pm 1$;
  Update a_μ and a_0 for the new state $\mathbf{X}$;
end

Algorithm 4: The Gillespie algorithm for stochastic dynamics. Because the process is a continuous time Markov chain the time in between the reactions varies, thereby bringing about the need to store the reaction timestamps. The choice of x_μ plus or minus one depends on if the decay or creation reaction was drawn.

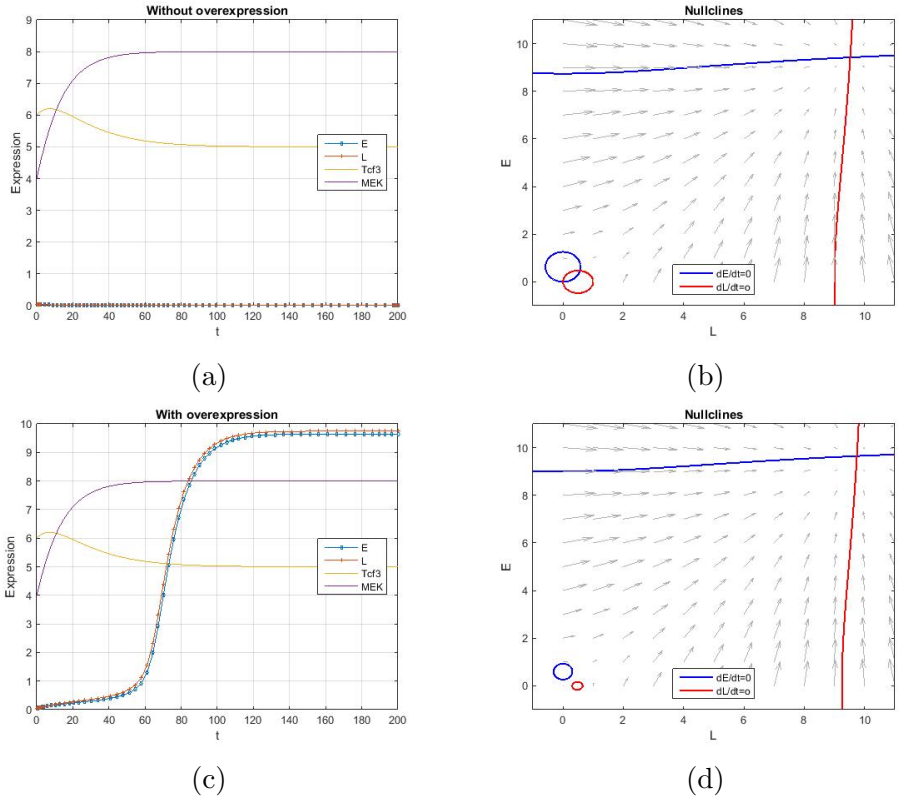

Figure 16: Deterministic dynamic results of the network in Figure 15. (a) A stable MEF state with low expression of early (E) and late (L) pluripotency genes without overexpression. (b) Nullclines of the system without overexpression. The lines cross at stationary states which show both a stable low-expression MEF state and a stable high-expression iPS state. (c) Transition from MEF to the iPS state after turning overexpression on. (d) Nullclines for the system with overexpression on demonstrating the loss of stability of the MEF-state in the lower left corner. In this particular network, the expression levels of the early and late genes are virtually indistinguishable, which confirms an unsuitable topology.

E Scaling Discrepancy of Stochastic and Deterministic Dynamical Simulations

An important difference between the stochastic and deterministic simulations is that the sccale of the stochastic simulations of the Gillespie algorithm is no longer arbitrary because the algorithm simulates individual reactions. In contrast, the deterministic solutions to the ODEs are in terms of of concentration levels which are normalized in relation to a specific housekeeping gene. This means that the optimized parameters might not find themselves suitable in the stochastic context. For instance, the MEF state can be stochastically unstable even though it's deterministically stable if slight perturbations cause the system to exit the state stochastically. From a biological perspective, the MEF state ought to be extremely stable because MEF cells do not spontaneously reprogram from just being plated in culture conditions. In other words,

$$\lim_{t\to\infty}([E]_t + [L]_t) \sim 0 \tag{E.26}$$

for a biologically feasible system without overexpression.

Consequently, the scale-dependence of the behavior of the network was investigated. A comparison between two different scales can be seen in Figure 17. In general, smaller scales tend to be less stable and more chaotic than larger ones (because the relevance of the production/decay of the reactants gets bigger as their total number decreases). On the other hand, a too large scale would be biologically irrelevant. Ideally, the scale would match the experimentally measured concentrations but the problem is that the experimental values are normalized in relation to a specific housekeeping gene.

In the end, the scale of subfigures (e-h) was chosen for the stochastic simulations because the standard deviation of the reactants quite closely matched experimental results.

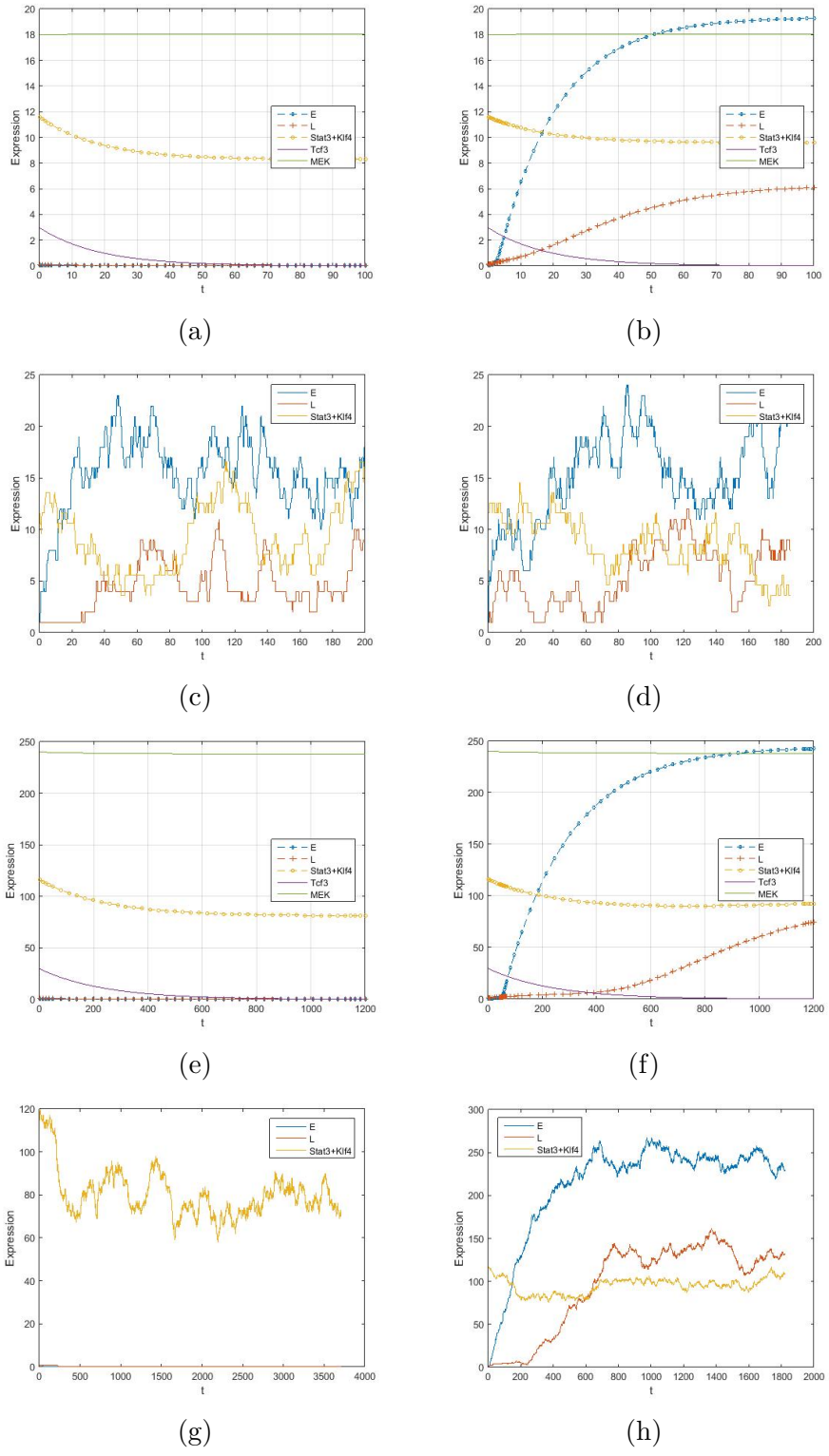

Figure 17: Deterministic and stochastic simulations of the network in Figure 10 with two different scales. E and L represents early and late pluripotency genes respectively. The first four figures (a-d) show low target concentrations and the last four figures (e-h) show high target concentrations. (a & e) deterministic stability of the MEF-state without overexpression. (b & f) deterministic iPS states. (c) Stochastic simulations of the non-overexpressed dynamics showing the system in an iPS-like state indicating a stochastically unstable MEF state. (g) stochastic stability of MEF at a higher scale. (d & h) Stochastic simulation of the overexpressed dynamics showing stable iPS states.

F Roadblock Investigation of Late Genes in the Minimal Network

This appendix covers the further investigation of the roadblock behavior of the late-node in the minimal network in hopes of identifying roadblocking culprits of the late genes. In this section, the late genes were separated from the late node one by one and looked at separately. If individually separated genes would inherit the large fluctuations while the rest of the late genes would not, that specific gene would likely be a roadblocking gene. Figure 18 presents the results. The results are summarized below together with some important remarks:

- Different optimizations were carried out for every new separation. The difference from the previous network is that there's an extra target time-series for the separated node. However, because all separated genes were from the late node, the target values for the optimization didn't change a lot between the different cases. This results in a lot of similarity in the figures.
- Separating Esrrb failed to produce a stochastically stable MEF state altogether, suggesting that this network is not realistic.
- The only gene that actually fluctuated more than the late node after separation was the one of the Nanog-separated network. This network did however not manage to reproduce the early and late behaviors of different genes.
- Separating Sox2 did not capture the late-like dynamics of Sox2, presumably because Sox2 is itself overexpressed, which forces it to a high value quickly.
- The separated nodes for Esrrb, Gbx2 and Klf2&5 all had lower fluctuations than the Late node, suggesting that these are not roadblocking genes.

In conclusion, Esrrb, Gbx2 and Klf2&5 were the only genes with lower fluctuations than the late node itself, possibly suggesting that these genes are not roadblocks. Nanog was the only gene which displayed larger fluctuations when separated from the late node. Unfortunately, the Nanog-separated simulations obliterate the early/late node behavior, making this particular model inapplicable as a model of real dynamics. The same loss of early/late behavior can be observed for the Sox2-separated simulations.

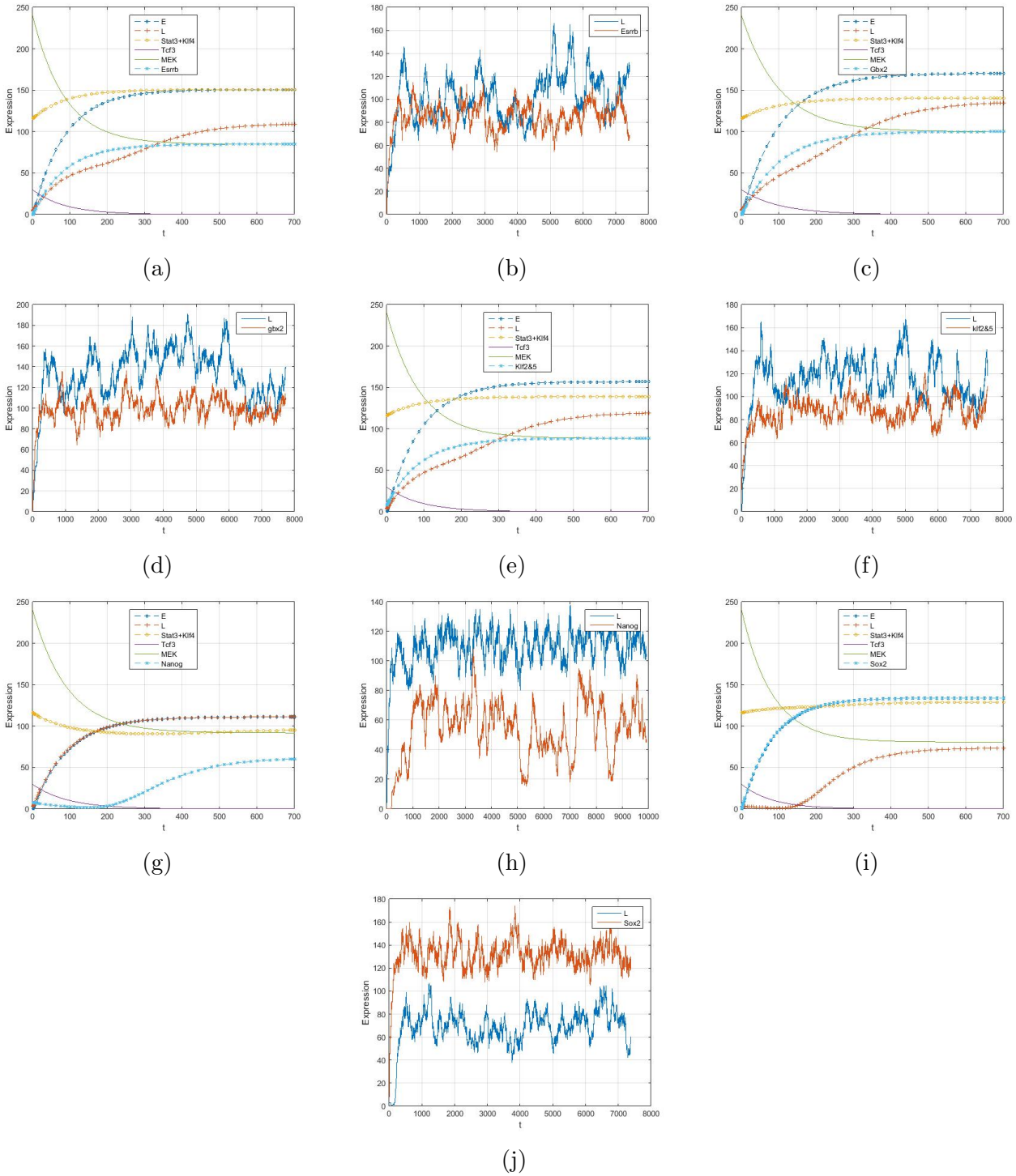

Figure 18: Deterministic and stochastic simulations after separating the Esrrb (a & b), Gbx2 (c & d), Klf2 together with Klf5 (e & f), Nanog (g & h) and Sox2 (i & j) nodes from the network in Figure 10. The purpose here is to uncover which genes are roadblocks by looking at their fluctuations compared to the late node itself. However, most of the networks demonstrate unrealistic behavior.

G The General Reverse Algorithm for Finding Basins of Attraction

The general reverse algorithm was applied early in the BN approach in this project in order to assess the validity of different BNs. However, the algorithm is extremely slow when calculating basins for states with very large basins (basins in our project could reach sizes of around 1.5 million states) and was therefore abandoned in favor of other quicker algorithms.

The algorithm was originally developed for random BNs in the form of cellular automata [44], but can be applied in the context of gene regulatory networks as well. In effect, the algorithm runs the network backwards in time. One setback with it was originally that backwards trajectories in general diverged for cellular automata, but fortunately this is not the case for gene regulatory networks.

The algorithm works in the following manner:

Input: $InitialState$, $TruthTable$, $ParentStates$, $CultureCondition$
Output: All pre-images of state $InitialState$
for *i=1* **to** *number of nodes*^{2} **do**
  remove all rows from $TruthTable(i)$ which do not agree with $InitialState$;
  remove all rows from $TruthTable(i)$ which are not of the current $CultureCondition$;
end
Create an empty array *Stack* of candidate pre-images;
Choose an initial node n;
for $i \subseteq$ *rows in* $TruthTable(i)$ **do**
  add the values of i to a new row of *Stack* in columns $ParentStates(n)$;
end
for *i=2* **to** *number of nodes* **do**
  for *j=1* **to** *length of Stack* **do**
    for *k=1* **to** *length of* $TruthTable(i)$ **do**
      if *row k of* $TruthTable(i)$ *is compatible with row j of Stack* **then**
        merge the rows into *Stack*;
      else
        continue;
      end
    end
  end
end

Algorithm 5: The General Reverse Algorithm for finding pre-images of a specific state. The algorithm returns all possible pre-images of state $InitialState$ as rows of the matrix *Stack*.

In short, the algorithm creates a stack of possible pre-images by considering the viable inputs for one additional node at a time; first two nodes together, then tree nodes together, then four, etc., until all nodes are considered. If the stack at any given point is reduced to zero, the input state has no pre-images – it is a garden state. To compute the basin, simply repeat the algorithm for all pre-images until none are left. Note that if *Stack* still contains any empty entries after the algorithm is done, these entries can attain all allowed value configurations. This is the case for leave-nodes because they have no feed-in edges to impact the next state.

Keep in mind that the order in which the nodes are considered is completely arbitrary. Wuensche stated in his original publication that for the most efficient computation, the order should correspond to the greatest overlap of wiring schemes. But in fact, we can increase the efficiency even further by clever arrangement of the nodes. Firstly, we establish that nodes which are being repressed impose more restrictions on the possible input values than those who are not. Secondly, we note that nodes with few parents are quicker to be fully considered than nodes with many parents (this may seem rather obvious, but in the original publication researching cellular automata all the nodes had the same number of parents). By considering these points, we managed to decrease the computation time roughly four-fold compared to only considering the overlap.

References

[1] Takahashi K, Yamanaka S. *Induction of Pluripotent Stem Cells from Mouse Embryonic and Adult Fibroblast Cultures by Defined Factors.* Cell 126:663–676, 2006.

[2] Wu DC, Boyd AS, Wood KJ. *Embryonic stem cell transplantation: potential applicability in cell replacement therapy and regenerative medicine.* Front Biosci. 12:4525–4235, 2007.

[3] Shinnawi R, Huber I, Maizels L, Shaheen N, Gepstein A, Arbel G, Tijsen A, Gepstein L. *Monitoring human-induced pluripotent stem cell-derived cardiomyocytes with genetically encoded calcium and voltage fluorescent reporters.* Stem Cell Reports. 5:582–596, 2015

[4] NIH Stem Cell Information Home Page. *Stem Cell Information* [Online]. 2016. //stemcells.nih.gov/info/basics/7.htm [May 16, 2018].

[5] Xu H, Baroukh C, Dannenfelser R, Chen E.Y., Tan C.M, Kou Y., Kim Y.E, Lemischka I.R, Ma'ayan A. *ESCAPE: database for integrating high-content published data collected from human and mouse embryonic stem cells.* Database 2013: Article ID:bat045, 2013.

[6] Kauffman SA. *The origins of order: self-organization and selection in evolution.* New York: Oxford University Press, 1993.

[7] Utikal J, Polo JM, Stadtfeld M, Maherali N, Kulalert W, Walsh RM, Khalil A, Rheinwald JG, Hochedlinger K. *Immortalization eliminates a roadblock during cellular reprogramming into iPS cells.* Nature 460:1145-1148, 2009.

[8] Britten RJ, Davidson EH. *Gene regulation for higher cells: a theory.* Science 165:349-357, 1969.

[9] Kaufmann S. *Homeostasis and Differentiation in Random Genetic Control Networks.* Nature 224:177–178, 1969.

[10] Xiao Y. *A Tutorial on Analysis and Simulation of Boolean Gene Regulatory Network Models.* Current Genomics 10:511-525, 2009,

[11] Novak B, Pataki Z, Ciliberto A, Tyson JJ. *Mathematical model of the cell division cycle of fission yeast.* Chaos 11:277-286, 2001.

[12] Bornholdt S. *Systems biology: less in more in model large genetic networks.* Science 310:449-451, 2005.

[13] Zhang YP, Qian M, Ouyang Q, Deng M, Li F, Tang C. *Stochastic model of yeast cell-cycle network.* Physica D 219:35-39, 2006.

[14] Nicolis G, Prigogine I. *Self-organization in nonequilibrium systems, from dissipative structures to order through fluctuations.* New York: Wiley, 1977.

[15] Albert R, Othmer HG. *The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster.* J Theor Biol. 223:1-18, 2003.

[16] Li F.T, Long T, Lu Y, Ouyang Q, Tang C. *The yeast cell-cycle network is robustly designed.* PNAS 101:4781-4786, 2004.

[17] Wang GY, Du C, Chen H, Simha R, Rong Y, Xiao Y, Zeng C. *Process-based network decomposition reveals backbone motif structure.* PNAS 107:10478-10483, 2010.

[18] Shmulevich I, Dougherty ER, Kim S, Zhang W. *Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks.* Bioinformatics 18:261–274, 2002.

[19] Huang S. *Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery.* J Mol Med 77:469-480, 1999.

[20] Shmulevich I, Gluhovsky I, Hashimoto RF, Dougherty ER, Zhang W. *Steady-state analysis of genetic regulatory networks modelled by probabilistic boolean networks.* Comp Funct Genomics 4:601-608, 2003.

[21] Brun M, Dougherty ER, Schmulevich I. *Steady-state probabilities for attractors in probabilistic Boolean networks.* Signal Processing 85:1993-2013, 2005.

[22] Ching WK, Zhang S, Ng MK, Akutsu T. *An approximation method for solving the steady-state probability distribution of probabilistic Boolean networks.* Bioinformatics 23:1511-1518, 2007.

[23] Cowles MK, Carlin BP. *Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review.* Journal of the American Statistical Association 91:883-904, 1996.

[24] Robert CP. *Convergence Control Methods for Markov Chain Monte Carlo Algorithms.* Statistical Science 10:231-253, 1995.

[25] Massey FJ. *The Kolmogorov-Smirnov Test for Goodness of Fit.* Journal of the American Statistical Association 46:68-78, 1951.

[26] Wei F, Dudley RM. *Two-sample Dvoretzky–Kiefer–Wolfowitz inequalities.* Statistics Probability Letters 82:636-644, 2012.

[27] Michaelis L, Menten ML. *Die Kinetik der Invertinwirkung.* Biochem Z. 49:333–369, 1913.

[28] Hill AV. *The possible effects of the aggregation of the molecules of hæmoglobin on its dissociation curves".* J. Physiol. 40:4-7, 1910.

[29] Shea MA, Ackers GK. *The OR control system of bacteriophage lambda. A physical-chemical model for gene regulation.* J Mol Biol. 181:211-30, 1985.

[30] Choi S. *Systems Biology for Signaling Networks.* New York: Springer-Verlag, 2010.

[31] Dunn SJ, Martello G., Yordanov B., Emmott S., Smith AG. *Defining an essential transcriptor factor program for naïve pluripotency.* Science 344:1056-1160, 2014.

[32] O'Malley J, Skylaki S, Iwabuchi KA, Chantzoura E, Ruetz T, Johnsson A, Tomlinson S.R, Linnarsson S, Kaji K. *High resolution analysis with novel cell-surface markers identifies routes to iPS cells.* Nature 499:88-91, 2013.

[33] Zhang P, Andrianakos R, Yang Y, Liu C. *Kruppel-like factor 4 (KLF4) prevents embryonic stem (ES) cell differentiation by regulating nanog gene expression.* JBC 285:9080-9189, 2010.

[34] Lyssiotis C.A, Foreman R.K, Staerk J, Garcia M, Mathur D, Markoulaki S, Hanna J, Lairson L.L, Charette B.D, Bouchez L.C, Bollong M, Kunick C, Brinker A, Cho C.Y, Schultz P.G, Jaenisch R. *Reprogramming of murine fibroblasts to induced pluripotent stem cells with chemical complementation of Klf4.* PNAS 106:8912-8917, 2009.

[35] Martello F, Bertone P, Smith A. *Identification of the missing pluripotency mediator downstream of leukaemia inhibitory factor.* The EMBO Journal 32:2561-2574, 2013.

[36] Yeo J.C, Jiang J, Tan Z.Y, Yim G.R, Ng J.H, Göke J, Kraus P, Liang H, Gonzales K.A.U, Chong H.C, Tan C.P, Lim Y.S, Tan N.S, Lufkin T, Ng H.H. *Klf2 Is an Essential Factor that Sustains Ground State Pluripotency.* Cell Stem Cell 14:864–872 , 2014.

[37] Martello G, Sugimoto T, Diamanti E, Joshi A, Hannah R, Ohtsuka S, Göttgens B, Niwa H, Smith A. *Esrrb Is a Pivotal Target of the Gsk3/Tcf3 Axis Regulating Embryonic Stem Cell Self-Renewal.* Cell Stem Cell 11:491-504, 2012.

[38] Cole M.F, Johnstone S.E, Newman J.J, Kagey M.H, Young R.A. *Tcf3 is an integral component of the core regulatory circuitry of embryonic stem cells.* Genes Dev 22:746-755, 2008.

[39] Müssel C, Hopfensitz M, Kestler HA. *BoolNet—an R package for generation, reconstruction and analysis of Boolean networks.* Bioinformatics 26:1378–1380. 2010.

[40] Hinkelmann F, Brandon M, Guang B, McNeill R, Blekherman G, Veliz-Cuba A, Laubenbacher R. *ADAM: Analysis of Discrete Models of Biological Systems Using Computer Algebra.* BMC Bioinformatics 12:295, 2011.

[41] Gonzalez Gonzalez A, Naldi A, Sánchez L, Thieffry D, Chaouiya C. *GINsim: A software suite for the qualitative modelling, simulation and analysis of regulatory networks.* Biosystemse 84:91-100, 2006.

[42] Gardner TS, Cantor CR, Collins JJ. *Construction of a genetic toggle switch in Escherichia coli.* Nature 403:339–342, 2000.

[43] Xu H, Ang Y, Sevilla A, Lemischka IR, Ma'ayan A. *Construction and Validation of a Regulatory Network for Pluripotency and Self-Renewal of mouse Embryonic Stem Cells.* PLOS Computational Biology 10:e1003777, 2014.

[44] Wuenche A. *The ghost in mahine: Basins of attraction of random Boolean networks.* Artificial Life III Proceedings, Santa Fe: Addison-Wesley, 1994.