Odefying Kirkham

Simon Johanning

August 31, 2016

Contents

1 Introduction **2**
1.1 Background 2
1.2 Kirkham paper 2
1.2.1 Motivation 2
1.2.2 Structure 3

2 Theory **3**
2.1 General approach to make discrete models continuous 4
2.1.1 HillCube 4

3 Practical implementation **6**
3.1 Investigation of the GRN as boolean network 6
3.1.1 Identification of stable BMs 6
3.1.2 Fixed point analysis of reported networks 8
3.2 Odefying Kirkham 10
3.2.1 Odefy 10
3.2.2 Practical approach to Odefy 10
3.2.3 Case study: negative regulation of Runx2 on Dlx5 . . 11
3.2.4 Discussion of case study 18
3.2.5 Practical aspects Odefy 20

4 Discussion **20**

Appendices **22**

1 Introduction

1.1 Background

This course work has been written in the context of the course "Modellierung biologischer und molekularer Systeme" (WS 2015/16) at the University of Leipzig.

It is based on a presentation of the paper *Early gene regulation of osteogenesis in embronic stem cells* by *G.R. Kirkham et. al.* (see [2]) in the context of aforementioned course.

It will lay out the basic ideas of the paper (1.2), give an overview of the mathematical ideas the methodology is based upon (2), and will discuss results from the investigation of the boolean networks, as well as their continuous homologues (3). It will close with a discussion of the paper and the derived results (4).

Since the course associated with this course work is about (mathematical) modeling, and the mathematical techniques used in the paper were merely sketched, this report will focus mostly on the mathematical ideas and modeling aspects employed in the paper, as well as the challenges in reproducing the results, and will only treat the biological context of [2] on the side.

1.2 Kirkham paper

1.2.1 Motivation

Motivated by the knowledge gap of networks regulating the differentiation of mouse embryonic stem cells into bone cells, the authors of [2] are interested in understanding the regulatory mechanisms of osteogenesis. Since mathematical models of gene regulatory networks (GRNs) have proven to characterize GRNs well, as well as making novel predictions, especially in the case of incomplete data, the authors investigate mathematical models suited for this application.

Since boolean models (BMs) capture the qualitative dynamics of biological systems in a number of studies, the authors chose to start out modeling the GRN under investigation with boolean models. In order to capture the quantitative dynamics of the system under investigation, they then transform these BMs into their continuous homologue functions in order to describe the system as a set of ordinary differential equations (ODE), a process call "odefication", using the Odefy toolbox.

This process is chosen since the description of a system through ODEs captures the dynamics of a system very well and helps the authors to identify a (unique) GRN, characterizing the influence of the transcription and growth factors under consideration.

1.2.2 Structure

As noted above, [2] investigates GRNs relevant to osteogenesis in embryonic stem (ES) cells. Based on existing research on osteogenetic differentiation of mouse ES cells, they derive a set of three genes and two growth factors (GFs) to be of particular interest for this GRN, and aim to identify the relationship between them. These are the genes Runx2, Dlx5, Msx2 and the GFs TGFβ1 and BMP2.

Consequently the authors test different GRNs involving these genes and GFs, narrowing down the possible candidates.

Starting from modeling the GRNs as Boolean Models, they transform these into differential equations ("odefication") fitted to measured data in order to test predictions about the GRNs behaviour under over- and under-expression of the GFs.

Their experimental data comprises a set of measurements of the TFs of interest (Dlx5, Msx2, Runx2) at 5 fixed time points, exposed to the GFs BMP2, TGFβ1 and a combination of these. The expression levels of the TFs were used to reduce the number of candidate GRNs through comparing the expression profiles to the stable states of the GRNs.

The remaining GRNs (i.e. the ones matching the experimental expression profiles) were then transformed into continuous ODEs, and their over- and underexpression behaviour was compared to experimental results, ruling out all but one network.

The paper closes with a discussion and the description of the experimental setup.

2 Theory

As mentioned in the Introduction, [2] starts out with boolean models[1] of the species in the GRN.

Boolean models (BMs) express qualitative biological knowledge and have shown quite fruitful for this purpose. Since however many approaches depend on a quantitative characterization of a system, the process of deriving

[1] For an elaboration on BM see 4

quantitative behaviour from a qualitative description is of utmost importance.

As mentioned in the introduction, [2] uses the process of 'odefication' in order to achieve a quantitative characterization of a system from a qualitative description (BM). Since [2] is very brief about the mathematical aspects of this process, this course work will in the following lay out the mathematical ideas for this approach (as described in [1] and [3]).

2.1 General approach to make discrete models continuous

In order to derive continuous models from the BMs B_i, continuous homologues of these have to be found. For this, the discrete variables x_i are replaced by continous variables $\overline{x_i} \in [0,1]$, yielding the *continuous homologue* functions $\overline{B_i} : [0,1]^{N_i} \to [0,1]$ on the unit interval.

Using the continous homologues of the BMs, the behavior of the species can be described through the differential equation

$\dot{\overline{x_i}} = \frac{1}{\tau_i}(\overline{B}_i(\overline{x}_{i_1}, \overline{x}_{i_2}, ..., \overline{x}_{i_{N_i}}) - \overline{x}_i)$,

where the production of $\overline{x}_i$ is given by $\overline{B}_i$, and $\tau_i := \frac{1}{\gamma_i}$ stands for the life-time of species X_i, with γ_i as the decay rate of X_i.

For the (in respect to the BM) homologuous continuous functions $\overline{B}_i$, three properties need to hold (as noted in [1]):

- **Accuracy**: $\overline{B_i}$ and B_i need to agree on the vertices of the unit cube $(\{0,1\}^{N_i})$ [2]
- **Good analytical properties** such as smoothness, so a mathematical analysis as a system of ODEs can be performed
- **Minimality and uniqueness**: $\overline{B}_i$ should be the unique minimal solutions in their interpolation class

2.1.1 HillCube

In order to achieve the desired properties sketched above, normalized *HillCubes* are constructed. HillCubes employ *Hill functions* on the edges of the unit cube described above for the functional behavior of the interpolations. As a first step to arrive at (normalized) HillCubes, *BooleCubes* are constructed. These are functions that interpolate B_i linearly through the use of multivariate polynomial interpolation. In order to get a unique solution,

[2] $\overline{B_i}$ are then called *perfect* continuous homologues, exhibiting similar steady-state behavior

the function with the minimal degree (of the polynomial) is chosen. This function satisfies the properties set out in 2.1.

The functions $\overline{B}_i^I(\overline{x}_{i_1}, \overline{x}_{i_2}, ..., \overline{x}_{i_{N_i}})$ define a system of ODEs that describe the temporal development of $\overline{x}_i$ through the equations

$\dot{\overline{x}}_i = \frac{1}{\tau_i}(\overline{B}_i^I(\overline{x}_{i_1}, \overline{x}_{i_2}, ..., \overline{x}_{i_{N_i}}) - \overline{x}_i)$.

These *BoolCube* functions are affine multilinear [3]. However, molecular interactions exhibit switch-like behavior rather than affine multilinear behavior. In order to model this, (sigmoid shape) *Hill functions* ($f(\overline{x}) = \frac{\overline{x}^n}{\overline{x}^n + k^n}$) are used.

These functions are parameterized by the *Hill coefficient* (n), that determines the slope of the curve and measures the cooperativity of the interaction, and the threshold where a species is considered as 'on' in the BM (k), thus where the value of activation is half maximal. Mathematically, the HillCube functions $\overline{B_i^H}$ are described through

$\overline{B_i^H}(\overline{x}_{i_1}, \overline{x}_{i_2}, ..., \overline{x}_{i_{N_i}}) := \overline{B}_i^I(f_{i_1}(\overline{x}_{i_1}), f_{i_2}(\overline{x}_{i_2}), ..., f_{i_n}(\overline{x}_{i_{N_i}}))$.

Since the Hill functions approach 1 only asymptotically, HillCubes are not perfect homologues of B_i, and not all desired properties sketched in 2.1 are fulfillled. Thus, in the odefy-approach, the Hill functions are subsequently normalized on the unit interval. HillCubes $\overline{B}_i^{H_n}$ are defined by $\overline{B}_i^{H_n}(\overline{x}_{i_1}, \overline{x}_{i_2}, ..., \overline{x}_{i_{N_i}}) := \overline{B}_i^I(\frac{f_{i_1}(\overline{x}_{i_1})}{f_{i_1}(1)}, \frac{f_{i_2}(\overline{x}_{i_2})}{f_{i_2}(1)}, ..., \frac{f_{i_{N_i}}(\overline{x}_{i_{N_i}})}{f_{i_{N_i}}(1)})$.

The value of the Hill function f_{i_j} on the value 1 thus normalizes the HillCubes on the unit interval. Normalized HillCubes yield a perfect continuous homologue of the Boolean functions B_i. Thus, a steady-state of the BM will also be a steady-state of the continuous system [4], as shown in [1]. However, the authors note that this method does not accurately transform the Boolean update rule into a continuous activation function.

This is due to a systematic difference between the Boolean logic and the analytic form of the activation function (see [1]). For the HillCube models, the imperfect agreement is caused by the asymptotic behavior of the Hill functions, and thus can be made arbitrarily small (through the appropriate choice of parameters), justifying the described approach.

[3] $1 \leq j \leq N_i, \overline{x}_{ik}, k \neq j$ fixed: $\exists a, b \in \mathbb{R} : \overline{B_i}^I(\overline{x}_{i_1}, \overline{x}_{i_2}, ..., \overline{x}_{i_{N_i}}) = a + b\overline{x}_{ij}$

[4] This exemplifies the importance of normalization, since steady-states of the BM are not (necessarily) steady-states of the non-normalized HillCubes.

3 Practical implementation

The investigation of the practical implementation of the discussed paper is twofold; Since this course work focuses on the mathematical modelling of GRN as boolean models, an investigation of the GRNs discussed in [2] will be performed in 3.1.

On the other hand, another focus of this course work is to follow the method employed in [2] and to investigate whether the same results can be reconstructed through the use of the odefy toolbox (see 3.2).

Despite the original plan to focus this course work mostly on the mathematical aspects of the approach of [2], in the practical investigation some inconsistencies, as well as undiscussed methodological ambiguities arose.

In order to overcome these, the author employed different approaches as well as different combinations of assumption to derive what method was used exactly. Due to this, the practical implementation took up a lot more space than was originally planned.

3.1 Investigation of the GRN as boolean network

In the spirit of [2], who used *"Boolean models [...] to identify networks that are consistent with the observed expression profiles [...]"*, the first step in this practical investigation was to identify these BMs myself (3.1.1), as well as performing a fixed point analysis on the BMs reported in [2] (see sssec:FPanalysis). These two approaches will be discussed in the following.

3.1.1 Identification of stable BMs

The first step of the practical investigation of the boolean networks was to find stable boolean networks that were consistent with the expression profiles measured in [2].

For this purpose, the author wrote a simple (Java) program that would calculate stable models for the GRN candidate networks (see Appendix B).

The program investigates stable expression profiles based on the GRNs as noted in [2], shown in figure 1.

In order to identify which of the GRNs in figure 1 yield stable models that correspond to the expression profiles (see figure 2), the author did some experiments with simple Java programs.

The preliminary studies of the authors led to the suspicion that the (adoption of the) methodology sketched in [2] is not quite consistent with the data presented in figure 2. Thus, the author wrote a program for a

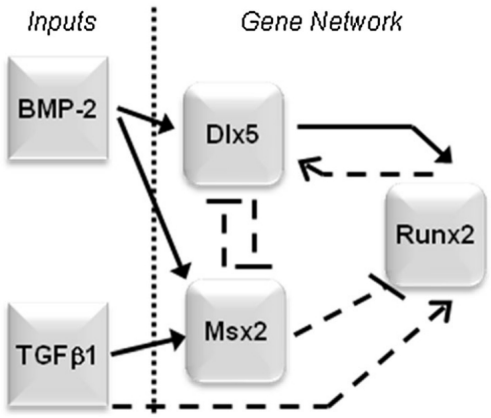

Figure 1: Potential regulatory networks derived from the literature, as noted in [2]

Time (hours)	Media	Gene expression		
		Dlx5	Msx2	Runx2
0		Off	Off	Off
24	BMP2	On	On	On
	TGFβ1	Off	Off	On
	BMP2/TGFβ1	On	On	Off

Figure 2: (experimentally) measured expression profiles of species Dlx5, Msx2 and Runx2 in media BMP2, TGFβ1 and BMP2 as well as TGFβ1 at the initial time and after 1 day (source: [2])

systematic study how many of the potential GRNs are consistent with the expression table in [2].

Of the $3^5 = 243$ potential GRNs in figure 1, [2] identified three that were stable and that corresponded to the expression data. The program the author wrote (which can be found in Appendix B) however found not a single one (indicated however that 24 (respective 36 with disjunctive logical connection) of these were not stable).

To get a more comprehensive understanding of the methodology employed in [2], in the second step, a fixed point analysis of the fitting model identified in the paper was performed. Since the methodology was not entirely clear from the paper, different assumptions were tested based on the

models identified by [2].

3.1.2 Fixed point analysis of reported networks

Since the process discussed in 3.1.1 yielded results inconsistent with the results discussed in [2], the logical step seemed to be a fixed point analysis of these GRNs.

As [2] reported that the only stable GRNs consistent with the (qualitative) expression profile measured (see figure 2) were the GRNs where the influence of TGFβ1 and Msx2 on Runx2, as well of Dlx5 on Msx2 were inhibitory and the influence of Msx2 on Dlx5 was activitory, a fixed point analysis was only done for the GRNs depicted in figure 3.

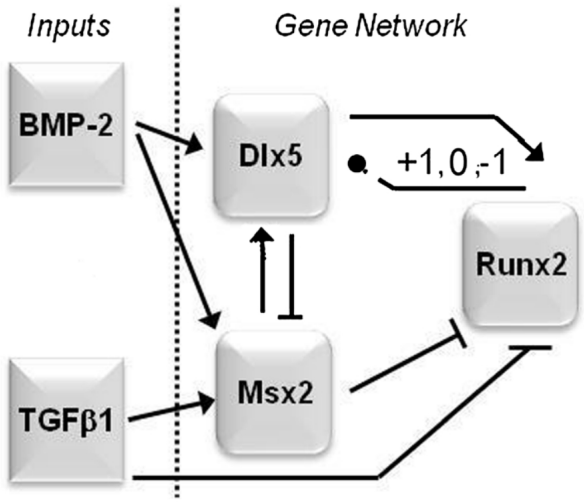

Figure 3: regulatory networks fitting expression profiles (source: [2])

Based on the inconsistency of the results of the first step sketched in 3.1.1 with [2], and the fact that the latter was not very clear about their assumptions, the goal of the fixed point analysis was to reconstruct the BMs used in the paper.

Since it was not entirely clear whether the logical connection in the boolean expression was disjunctive or conjunctive, as well as how the absence of a GF in the medium was modelled in the boolean expression, different combinations were tried out for the fixed point analysis with the hope to derive which boolean expressions [2] was based upon.

In the following, a fixed point analysis based on figure 3 will be performed, where the influence of Runx2 on Dlx5 will be activatory (case 1), inhibitory (case 2) or no influence of Runx2 on Dlx5 will be assumed (case 3). Each case itself will contain three subcase specifying the environment (x.1: only BMP2 present, x.2: only TGFβ1 present, x.3: BMP2 as well as TGFβ1 present).

Since [2] is not explicit on whether the boolean expressions are linked by conjunctions or disjunctions, both cases will be investigated, and in the best case the results will indicate whether the junction is conjunctive or disjunctive.

Due to the large number of cases these missing assumptions generate, the 36 fixpoint analyses can be found in the appendix (Appendix C).

An analysis of these cases shows that none of the assumptions are consistent with the boolean networks reported in [2]. For the case where only TGFβ1 is present, none of the assumptions generate a steady state where both Dlx5 and Msx2 are not expressed and Runx2 is expressed.

This expression pattern is unexpected for the regulatory networks depicted in figure 3 since TGFβ1 has an inhibitory effect on Runx2 and thus the expression of Runx2 would only be expected with an expression of Dlx5. figure 2 however shows that a low expression of Dlx5 was measured. Thus this regulatory network seemed surprising to the intuition of the author.

The case of the GRN with the presence of TGFβ1 however is not the only case when the steady state of the boolean network does not correspond to the reported expression profile. In the case of the influence of Runx2 on Dlx5 to be activatory and both GFs to be present, none of the assumptions reproduce the GRNs measured behaviour that both Dlx5 and Msx2 are expressed but Runx2 is not.

This case seems a little more intuitive; however if a conjunctive connection of the expressions in the boolean terms is assumed, only expression of Msx2 is reached (since Dlx5 will not be expressed as Runx2 is not expressed). If a disjunctive connection in the boolean terms is assumed, all three species will be expressed (since Runx2 will be expressed when Dlx5 is expressed).

In the case of inhibitory or no influence of Runx2 on Dlx5 with the presence of both GFs, an oscillation is observed for the conjunctive case. This occurs since the expression of Msx2 leads to the expression of Dlx5, which however leads to the inhibition of the expression of Msx2. With the

lack of expression of Msx2, Dlx5 will not be expressed anymore and not inhibit the expression of Msx2 anymore, leading to it being expressed again.

figure 2 however sees the expression of both species as a steady state.

3.2 Odefying Kirkham

Contrary to 3.1, this subsection will discuss the attempt at reconstructing the results presented in [2] through the use of the Odefy 1.19 toolbox in GNU Octave 4.0.2. Since Odefy was written for octave 3.x, some minor changes to the software were necessary (see 3.2.5).

In order to follow the workflow of the study, first the candidate networks depicted in figure 1 were created. Subsequently the corresponding continuous models were created (Appendix D), before their behaviour over time was simulated (Appendix D).

For the interesting case [2] reports (that conforms to their measured data), a more extensive case study is done in 3.2.3, whose results will be discussed in 3.2.4.

3.2.1 Odefy

Odefy, a MATLAB and GNU Octave compatible toolbox for the "odefication" (transformation into continous ODEs) of boolean models, has been developed by the Institute for Computational Biology of the Helmholtz-Zentrum München.

As noted in 1.2.1, ODEs allow for a more detailled and quantitative characterization of GRNs. Qualitative (biological) knowledge is represented as BMs, which are transformed into continuous model, that are fitted with quantitative data (for a sketch of the workflow see figure 4).

Odefy uses the HillCube modeling technique. With $\overline{B}_i$ as the HillCube model of $\overline{x}_i$, the following differential equation is reached: $\dot{\overline{x}}_i = \frac{1}{\tau_i}(\overline{B}_i(\overline{x}_{i_1}, \overline{x}_{i_2}, ..., \overline{x}_{i_{N_i}}) - \overline{x}_i)$, where τ_i stands for the lifespan of species X_i (as described in 2.1).

3.2.2 Practical approach to Odefy

In Odefy, the boolean models are specified as a set of strings describing the influences of other species onto the species specified. The semantics of these strings of the form $LHS = RHS$ is that LHS is considered 'on' iff RHS is true, and 'off' otherwise.

The candidate BMs discussed in [2], as derived from the literature (see also figure 1), use the GFs BMP2 and TGFβ1, as well as the genes Dlx5,

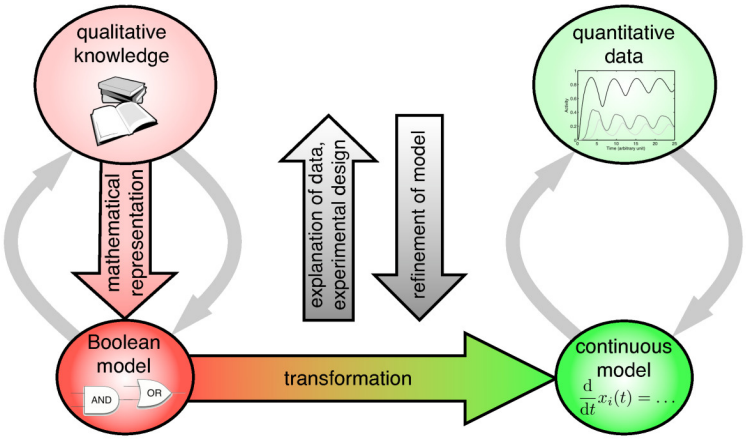

Figure 4: Sketch of the odefy workflow.

Msx2 and Runx2. Since the authors assumed that the influence of TGFβ1 to Msx2, BMP2 to Msx2 and Dlx5, as well as the influence of Dlx5 on Runx2 was activatory, the following substrings are fixed in the model description (odefy-syntax):

'Msx2 = TGFβ1 && MBP2'[5], 'Dlx5 = BMP2', 'Runx2 = Dlx5'

The approach in this course work is two-fold; it generates the potential candidate networks as noted in figure 1 and performs simulations of (hill-cube) continuous homologue. Due to the large number of resulting networks ($2 * 3^5$), the results will not be discussed here, and can be found in 4. This section will instead focus on the reduced set of candidate networks in [2], where the influence of TGFβ1 and Msx2 on Runx2 is inhibitory, the influence of Msx2 on Dlx5 is excitatory and the influence of Dlx5 on Msx2 is inhibitory (see figure 3).

However, with the three growth media, two possible logical connections and three candidate networks, only a representative case (where Runx2 regulates Dlx5 negatively[6]) is presented in the following.

3.2.3 Case study: negative regulation of Runx2 on Dlx5

As mentioned in 3.2.2, this case study will sketch the process from the boolean model to the continuous expression profile for the regulatory net-

[5] As discussed in 3.1 it is not quite clear whether the junction of the species is a conjunction or a disjunction. In the spirit of 4 in this investigation a conjunction is used.

[6] The case that [2] identifies as matching the experimentally observed oscillation behaviour

work where Runx2 regulates Dlx5 negatively.

As the boolean model is specified as a boolean expression, a boolean expression for the corresponding network is specified by

```
Runx2Dlx5Inhibitory_and = {
        'BMP2 = <>',
        'TGFb1 = <>',
        'Dlx5 = Msx2 && ~Runx2 && BMP2',
        'Msx2 = BMP2 && TGFb1 && ~Dlx5',
        'Runx2 = Dlx5 && ~Msx2 && ~TGFb1'
}
```

for the case that the logical connection is a conjunction, and

```
Runx2Dlx5Inhibitory_or = {
        'BMP2 = <>',
        'TGFb1 = <>',
        'Dlx5 = Msx2 || ~Runx2 || BMP2',
        'Msx2 = BMP2 || TGFb1 || ~Dlx5',
        'Runx2 = Dlx5 || ~Msx2 || ~TGFb1'
}
```

for the case the logical connection is a disjunction.

The corresponding boolean models and the structs needed for the simulation by Odefy are constructed as follows:

```
model_and = ExpressionsToOdefy(Runx2Dlx5Inhibitory_and)
simstruct_and = CreateSimstruct(model_and)
simstruct_and.type = 'hillcube'
model_or = ExpressionsToOdefy(Runx2Dlx5Inhibitory_or)
simstruct_or = CreateSimstruct(model_or)
simstruct_or.type = 'hillcube'
```

under the usage of the HillCube method as discusssed in 2.1.1.

Since in both cases, the expression of the genes under investigation was considered off at the beginning of the simulation[7], the initial values for these species was set to 0:

```
simstruct_and = SetInitialValue(simstruct_and, 'Dlx5', 0)
simstruct_and = SetInitialValue(simstruct_and, 'Msx2', 0)
simstruct_and = SetInitialValue(simstruct_and, 'Runx2', 0)
simstruct_or = SetInitialValue(simstruct_or, 'Dlx5', 0)
```

[7]Since as [2] notes, mRNA levels are low in mouse embryonic stem cells

```
simstruct_or = SetInitialValue(simstruct_or, 'Msx2', 0)
simstruct_or = SetInitialValue(simstruct_or, 'Runx2', 0)
```

For the individual cases (presence of media / logical connection in boolean expression), the following results were obtained:

BMP2 only with conjunction With this setup, the following code is executed:

```
simstruct_and = SetInitialValue(simstruct_and, 'BMP2', 1)
simstruct_and = SetInitialValue(simstruct_and, 'TGFb1', 0)
[t,y] = OdefySimulation(simstruct_and, 0, 0);
plot(t,y, {".", ".", 'd', '*', 'p'}, "markersize", 2);
legend(simstruct_and.model.species);
xlabel("time");
ylabel("expression");
title("Runx2->Dlx5 inhibitory, BMP2 only, conjunction");
axis([0, 10, -0.1, 1.1]);
```

The simulation yields a static behaviour where no dynamic is visible (as can be seen in figure 5)

TGFb1 only with conjunction With this setup, the following code is executed:

```
simstruct_and = SetInitialValue(simstruct_and, 'BMP2', 0)
simstruct_and = SetInitialValue(simstruct_and, 'TGFb1', 1)
[t,y] = OdefySimulation(simstruct_and, 0, 0);
plot(t,y, {".", ".", 'd', '*', 'p'}, "markersize", 2);
legend(simstruct_and.model.species);
xlabel("time");
ylabel("expression");
title("Runx2->Dlx5 inhibitory, TGFb1 only, conjunction");
axis([0, 10, -0.1, 1.1]);
```

The simulation yields a static behaviour where no dynamic is visible (as can be seen in figure 6)

TGFb1 and BMP2 with conjunction With this setup, the following code is executed:

```
simstruct_and = SetInitialValue(simstruct_and, 'BMP2', 1)
simstruct_and = SetInitialValue(simstruct_and, 'TGFb1', 1)
```

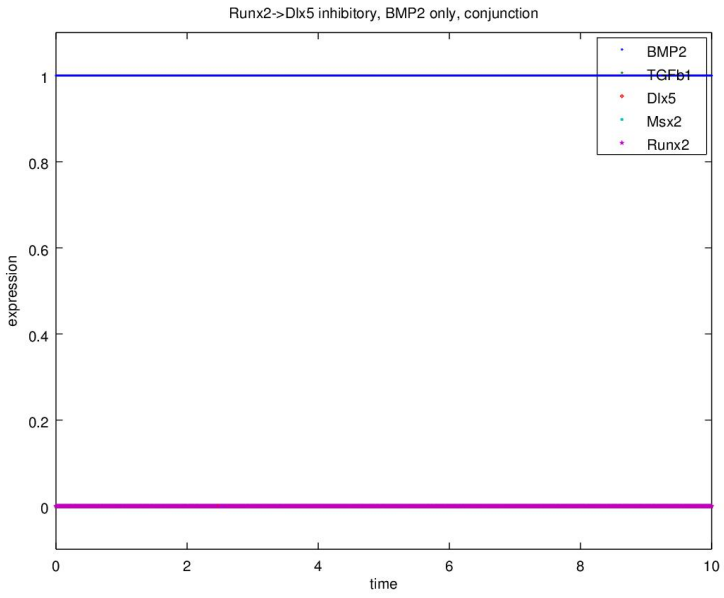

Figure 5: conjunctive logical connection of boolean expressions with the presence of the GF BMP2

```
[t,y] = OdefySimulation(simstruct_and, 0, 0);
plot(t,y, {".", ".", 'd', '*', 'p'}, "markersize", 2);
legend(simstruct_and.model.species);
xlabel("time");
ylabel("expression");
title("Runx2->Dlx5 inhibitory, TGFb1 and BMP2, conjunction");
axis([0, 10, -0.1, 1.1]);
```

With the presence of both growth factors, dynamics can be observed in all three species (as can be seen in figure 7). The activity of Runx2 is negligibly small; the expression levels of Dlx5 and Msx2 however express a clear dynamic after some (rudimentary) transient oscillation, when they stabilize at roughly half the maximal expression level.

BMP2 with disjunction With this setup, the following code is executed:

```
simstruct_or = SetInitialValue(simstruct_or, 'BMP2', 1)
simstruct_or = SetInitialValue(simstruct_or, 'TGFb1', 0)
[t,y] = OdefySimulation(simstruct_or, 0, 0);
```

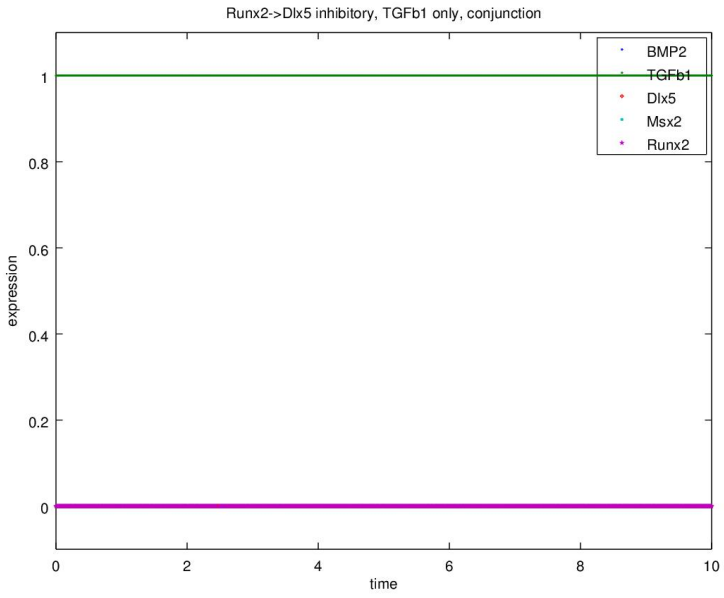

Figure 6: conjunctive logical connection of boolean expressions with the presence of the GF TGFb1

```
plot(t,y, {".", ".", 'd', '*', 'p'}, "markersize", 2);
legend(simstruct_or.model.species);
xlabel("time");
ylabel("expression");
title("Runx2->Dlx5 inhibitory, BMP2 only, disjunction");
axis([0, 10, -0.1, 1.1]);
```

In the case of disjunctive logical connection in the boolean expression and the presence of the GF BMP2, the species exhibit a monotonous increase in concentration with seemingly logarithmic growth (see figure 8). Although the different species exhibit similar qualitative behaviour, a clear difference in expression levels can be seen between the species with expression levels of Msx2 relatively low compared to the other two species and only Runx2 being completely expressed.

TGFb1 with disjunction With this setup, the following code is executed:

```
simstruct_or = SetInitialValue(simstruct_or, 'BMP2', 0)
```

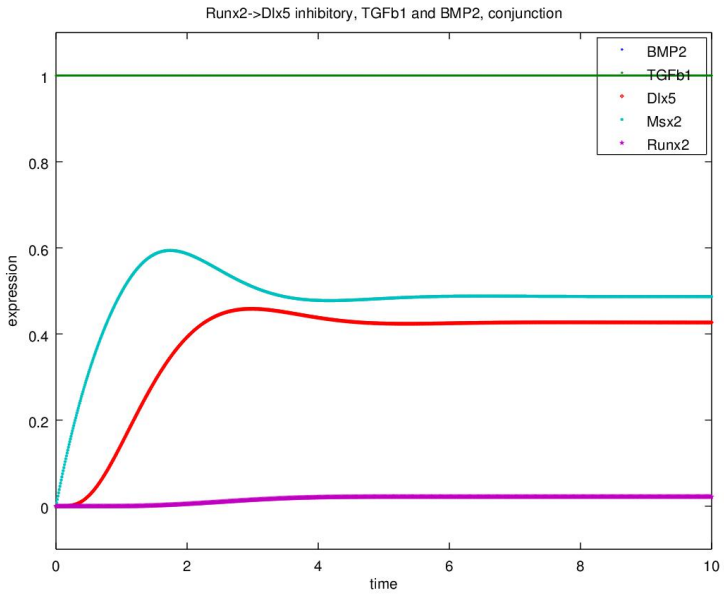

Figure 7: conjunctive logical connection of boolean expressions with the presence of both GFs

```
simstruct_or = SetInitialValue(simstruct_or, 'TGFb1', 1)
[t,y] = OdefySimulation(simstruct_or, 0, 0);
plot(t,y, {".", ".", 'd', '*', 'p'}, "markersize", 2);
legend(simstruct_or.model.species);
xlabel("time");
ylabel("expression");
title("Runx2->Dlx5 inhibitory, TGFb1 only, disjunction");
axis([0, 10, -0.1, 1.1]);
```

The disjunctive BM with the presence of TGFβ1 exhibits a similar behaviour as the medium of BMP2 with a monotonous increase in expression levels with seemingly logarithmic growth (see figure 9). In contrast to the medium with BMP2 however, the expression levels differ. In this medium, the expression of Msx2 is the strongest of the three, and the other two species (Dlx5 and Runx2) are comparatively weekly expressed.

Both media with disjunction With this setup, the following code is executed:

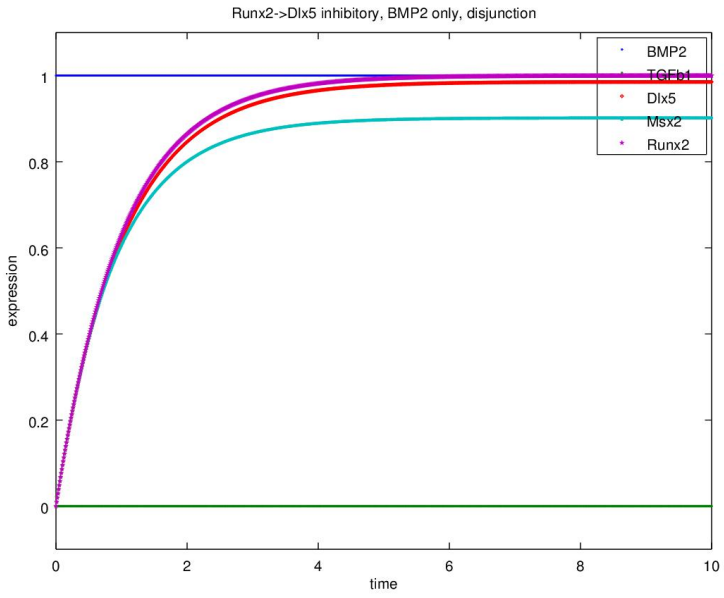

Figure 8: disjunctive logical connection of boolean expressions with the presence of BMP2

```
simstruct_or = SetInitialValue(simstruct_or, 'BMP2', 1)
simstruct_or = SetInitialValue(simstruct_or, 'TGFb1', 1)
[t,y] = OdefySimulation(simstruct_or, 0, 0);
plot(t,y, {".", ".", 'd', '*', 'p'}, "markersize", 2);
legend(simstruct_or.model.species);
xlabel("time");
ylabel("expression");
title("Runx2->Dlx5 inhibitory, TGFb1 and BMP2, disjunction");
axis([0, 10, -0.1, 1.1]);
```

The disjunctive BM with the presence of both media exhibits a similar behaviour as the individual media. All species exhibit a monotonous increase in expression levels with seemingly logarithmic growth (see figure 10). In this scenario, the expression of Runx2 is visibly weaker than the expression of Dlx5 and Msx2, which exhibit a very similar behaviour approaching almost full expression before they stagnate.

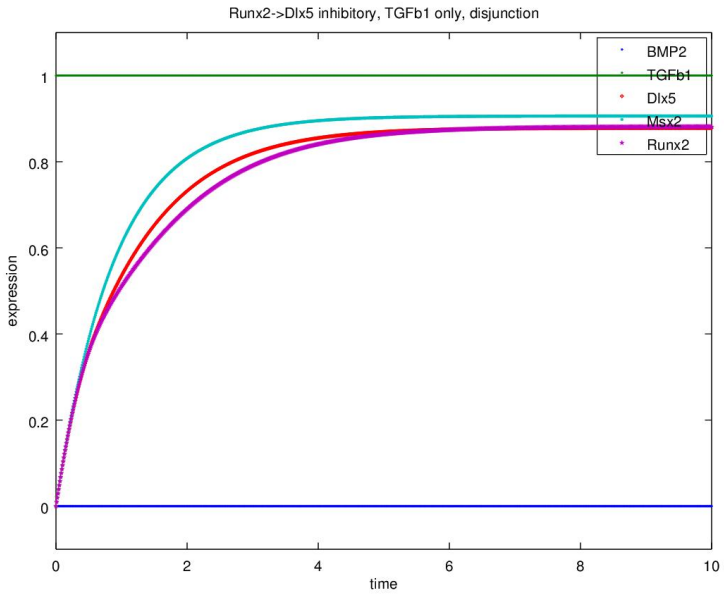

Figure 9: disjunctive logical connection of boolean expressions with the presence of TGFb1

3.2.4 Discussion of case study

The case study (3.2.3) shows very clear differences in the expression profiles under the assumption of the logical connection within the boolean expression. The intuition that led to the investigation of the different connections and the resulting behaviour in the perfect homologues of the boolean models thus seems to be confirmed.

The qualitative behaviour of the expression profiles with the logical disjunction shows a clear increase in expression for all three species in all three media. Comparing this behaviour to the behaviour discussed in [2] however, shows that the logical connection employed in this was clearly not the disjunction, since with the presence of both media an oscillatory behaviour should be observed, and in the presence of just one media, the expression of all three species is expected to very quickly reach a stable level. With the behaviour reported in 3.2.3 for the disjunction however, a continuous increase of all three species is observed for all three media, describing a very different behaviour. It can thus be assumed that the disjunction was not the logical connection of choice for [2].

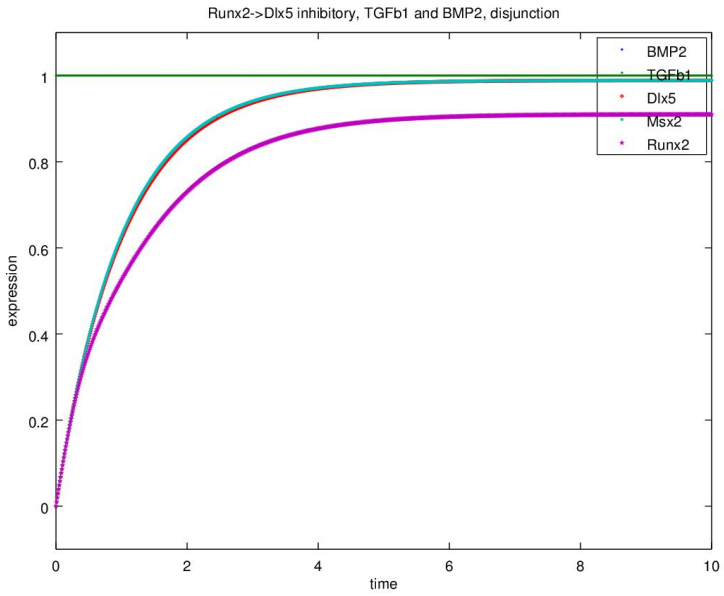

Figure 10: disjunctive logical connection of boolean expressions with the presence of TGFb1 and BMP2

The first three cases reported in 3.2.3 (in which a conjunction was employed in the boolean expressions) however are more interesting, since a more varied and qualitatively similar behaviour to [2] was observed. In the presence of only one of the GFs the (conjunctive) perfect homologue shows stationary behaviour, similar to the (qualitative) behaviour in [2]; however in their data, transient oscillations (as short as they may be) are present, whereas in this course work, these weren't reproducible. In the case of the presence of both GFs however, the differences are even more pronounced; Although some (slight) oscillatory behaviour can be observed in figure 7 for Dlx5 and Msx2, this behaviour seems more like transient oscillations than proper oscillations (for all three species), as reported in [2].

In summary, it can be said that although some tendencies of the desired behaviour were reproduced in this course work, this course work failed to reproduce the behaviour reported in [2]. However, these investigations illuminated the question whether a conjunctive or disjunctive logical connection was used for the boolean model in [2].

3.2.5 Practical aspects Odefy

Since Odefy 1.19 (at the time of writing the most current version of the odefy toolbox) was written with GNU Octave 3.x in mind, a few minor changes were necessary in order to make the toolbox work for Octave 4.0.2 (at the time of writing the most current version of GNU octave).

The author chose to include the current version of Octave in the setup, since he assumes that most (future) readers will use this version, and the following paragraphs would enable them to reproduce the discussed results as well as continue using Odefy (unless a future release of Odefy will take these into account).

The (minor) changes are as follows:
code/simulation/OdefySimulation.m:
The *extname* must not be in the *cmd* variable, thus line 80 needs to be changed to:

```
cmd = sprintf('yp = lsode(@(t,y)%s(y,t,simparams),
simstruct.initial'', realtime);',tmpname);
```

code/models/validvarname.m:

Another condition to set *isvalid* to 0 must be included (appended in line 16):

```
if numel(regexp('ydw', '[^A-Za-z_0-9]'))>0
    isvalid=0;
end
```

The behaviour for replacing invalid characters by underscore need to be modified. As such lines 27-32 need to be replaced by:

```
varname=regexprep('ydw*?', '[^A-Za-z_0-9]', '_');
```

4 Discussion

The paper analyzed in this course work employed a promising approach to identifying gene regulatory networks from a modelling perspective.

It was light on the mathematical background, which could be found easily (and well elaborated) in the papers [3] and [1].

Its brevity on the employed methodology however has shown to lead to numerous challenges in reproducing the results reported. Avoiding to state the assumptions the boolean models are based upon have led to a large number of cases and variations identified in 3.1.2.

However, even with the large number of cases or possible combinations of assumptions, no consistent results on the BM used in [2] could be found.

None of the BM reported to be consistent with the expression profiles listed in figure 2 were confirmed in 3.1.2, despite testing a rather large set of assumptions. This indicates that either the methodology used in the paper was not reported sufficiently or the inconsistencies arising during the research were resolved differently than in this course work. Most certainly the author does not want imply that the discrepancies to [2] couldn't be attributed to his lack of background in this field, and would most humbly request for clarification in methodology.

The oscillatory behaviour of the GRN reported by [2] to be consistent with the data reported in figure 2 was not originally meant to be part of this course work. However, due to the inconclusiveness of 3.1.2, the author hoped to gain an insight into the formulation of the boolean expressions used in [2]. Despite some major differences in the behaviour after transient oscillation, this comparison implies that the logical connection in the boolean expressions were the conjunction (which is consistent with Appendix A).

Resumptively, although intrigued by the approach sketched in [1] and [3], the author does not share the intuition that boolean models are suited to describe the behaviour of GRNs in more general contexts, and has not yet been convinced that they are suited for the research described here. Although interesting and promising, the author is left sceptical of the results presented in [2], which however could be due to his lack of training in its methodology.

Despite the broad approach taken in this course work, no results consistent with [2] could be achieved, and not even a clear result could be formulated as to what assumptions the boolean models investigated in [2] were based upon.

Appendices

Appendix A: Boolean models

This appendix will describe the propaedeutic aspects of boolean models not discussed in the course work.

As mentioned in 2, Boolean models (BMs) express qualitative biological knowledge, meaning that a species is either active or inactive (binary state). Represented as a graph (see figure 1), the species are seen as nodes, whereas the interactions between the species are modeled as edges (which can be either activating, inhibatory or no influence). Despite their coarseness, BMs can reproduce the qualitative behaviour of biological systems.

Due to their discrete nature however, they are not suited for quantitative models. Thus they can neither explain nor predict quantitative experiments, which is increasingly important in systems biology. For these applications, the transformation to systems of ordinary differential equations (ODEs) has shown to be fruitful.

Formal description

Formally, boolean models for N species $X_1, X_2, ..., X_N$, represented by variables $x_i \in \{0,1\}$, are defined by a set of species $R_i := \{X_{i_1}, X_{i_2}, ..., X_{i_{N_i}}\} \subset \{X_1, ..., X_N\}$ influencing X_i, as well as an update function $B_i : \{0,1\}^{N_i} \to \{0,1\}$ for every combination of $(x_{i_1}, ..., x_{i_2}, x_{i_{N_i}}) \in \{0,1\}^{N_i}$.

Mapped on the (hyper-) unit cube, the nodes $(\xi_{i_1}, \xi_{i_2}, ..., \xi_{i_{N_i}}) \in \{0,1\}^{N_i}$ corresponding to the domain of B_i, can logically be interpreted as $(\bigwedge_{ij|\xi_{ij}=1} x_{ij}) \wedge (\bigwedge_{ij|\xi_{ij}=0} \neg x_{ij})$, meaning that B_i will be true for the nodes of the cube where the coordinate equals the boolean value of influence of the activation function. With the sum-of-product representation, B_i can now be viewed as: $B_i(x_{i_1}, x_{i_2}, ..., x_{i_{N_i}}) = \bigvee_{(\xi_{i_1}, \xi_{i_2}, ..., \xi_{i_{N_i}})|B_i(\xi_{i_1}, \xi_{i_2}, ..., \xi_{i_{N_i}})=1} [\ (\bigwedge_{ij|\xi_{ij}=1} x_{ij}) \wedge (\bigwedge_{ij|\xi_{ij}=0} \neg x_{ij})]$.

Thus each product can be represented as a hyperedge between the start nodes $S \subset \{X_{i_1}, X_{i_2}, ..., X_{i_{N_i}}\}$ and the end node X_i, with each pair $(s, X_i), s \in S$ carrying a sign on whether it is noted as a factor x_{ij} or $\neg x_{ij}$ in the BM.

Appendix B: Code identification of stable BMs

This appendix will list the code used to identify stable boolean models from the GRNs (see 3.1.1) proposed in [2].

```
/**
 * Created by Simon Johanning for the course work "
    Odefying Kirkham"
 */
public class Main {
    //Constant to indicate whether the boolean
       junctions are conjunctions (AND) or
       disjunctions (OR)
    private static final String BOOLMODE = "OR";
    //number of maximal iterations to detect
       oscillating models (shouldn't arise though)
    private static final int MAXITERATIONS = 100000;

    private static int numIndexMaxIt = 0;
    private static int numFittingIndices = 0;

    public static void main(String[] args){
        //iterate through all BNs, encoded as
           elaborated in the code documentation
        for(int index=0;index<243;index++){
            if(testIndexInMedia(index)){
                printMatrix(fillMatrix(index));
                numFittingIndices++;
            //print BN if all matrices should be
               printed
            }else if(PRINTALLMATRICES){
                printMatrix(fillMatrix(index));
            }
        }
        System.out.println("Number of matrices with
           too many iterations: "+numIndexMaxIt);
        System.out.println("Number of fitting matrices
           : "+numFittingIndices);
    }

    /**
     * Test whether BN fits experimental data in all
        relevant media (BMP2, TGFb1 or both)
     *
```

```
 * @param index the integer encoding the BN under
    consideration
 * @return boolean value indicating whether BN
    fits experimental data
 */
private static boolean testIndexInMedia(int index)
    {
    int BMP2, TGFb1;
    HashMap<String, Boolean> stableExpressionData;
    //test index for medium BMP2
    if(DEBUGFLAG) System.out.println("\n
       ----------BMP2 only medium
       ----------\n");
    BMP2 = 1;
    TGFb1 = 0;
    //get a stable expression profile (steady-
       state)
    stableExpressionData = stableExpressionProfile
       (BMP2, TGFb1, fillMatrix(index));
    //if expression profile doesn't fit data,
       reject it
    if(!stableExpressionProfileDataFit(intToBool(
       BMP2), intToBool(TGFb1),
       stableExpressionData)) return false;
    //text index for medium TGFb1
    if(DEBUGFLAG) System.out.println("\n
       ----------TGFb1 only medium
       ----------\n");
    BMP2 = 0;
    TGFb1 = 1;
    //get a stable expression profile (steady-
       state)
    stableExpressionData = stableExpressionProfile
       (BMP2, TGFb1, fillMatrix(index));
    //if expression profile doesn't fit data,
       reject it
    if(!stableExpressionProfileDataFit(intToBool(
       BMP2), intToBool(TGFb1),
       stableExpressionData)) return false;
    //test index for both media
```

```
        if(DEBUGFLAG) System.out.println("\n
            ----------both GFs medium
            ----------\n");
        BMP2 = 1;
        TGFb1 = 1;
        //get a stable expression profile (steady-
            state)
        stableExpressionData = stableExpressionProfile
            (BMP2, TGFb1, fillMatrix(index));
        //if expression profile doesn't fit data,
            reject it
        if(!stableExpressionProfileDataFit(intToBool(
            BMP2), intToBool(TGFb1),
            stableExpressionData)) return false;

        //since expression data has not been rejected,
             it fits the data in all three cases, and
            can be accepted
        return true;
    }

    /**
     * Function to test whether the expression data of
         the BN fits the experimental data in the
        specified medium.
     * Expression data fit is determined by table 1 in
         Kirkham et al. 2012
     *
     * @param BMP2 boolean to indicate whether BMP2 is
         present in the medium
     * @param TGFb1 boolean to indicate whether TGFb1
        is present in the medium
     * @param stableExpressionData expression data of
        the steady state of the BN under consideration
     * @return boolean indicating the expression data
        of the BN fits the experimental data in the
        specified medium
     * @throws IllegalArgumentException gets thrown
        when the combination of GFs in the medium is
        not allowed / not of interested (i.e. neither
```

```
    BMP2 nor TGFb1 is present in the medium)
 */
private static boolean
    stableExpressionProfileDataFit(boolean BMP2,
    boolean TGFb1, HashMap<String, Boolean>
    stableExpressionData) throws
    IllegalArgumentException{
     //if the medium under consideration contains
         both BMP2 and TFGb1
     if(BMP2 && TGFb1){
         //model fits the data in the BMP2/TGFb1
             environment if Dlx5 and Msx2 are
             expressed, but Runx2 is not
         if(!DEBUGFLAG) return ((
             stableExpressionData.get("Dlx5")) && (
             stableExpressionData.get("Msx2")) && !(
             stableExpressionData.get("Runx2")));
         else{
             System.out.println("\
                 nstableExpressionData of Dlx5 is "
                 +(stableExpressionData.get("Dlx5"))
                 +" and should be true in order to
                 fit");
             System.out.println("\
                 nstableExpressionData of Msx2 is "
                 +(stableExpressionData.get("Msx2"))
                 +" and should be true in order to
                 fit");
             System.out.println("\
                 nstableExpressionData of Runx2 is "
                 +(stableExpressionData.get("Runx2")
                 )+" and should be false in order to
                 fit");
             return ((stableExpressionData.get("
                 Dlx5")) && (stableExpressionData.
                 get("Msx2")) && !(
                 stableExpressionData.get("Runx2")))
                 ;
         }
     }
```

```
//if the medium under consideration contains
    only BMP2
else if(BMP2){
    //model fits the data in the BMP2
        environment if Dlx5, Runx2 and Msx2 are
         expressed
    if(!DEBUGFLAG) return ((
        stableExpressionData.get("Dlx5")) && (
        stableExpressionData.get("Msx2")) && (
        stableExpressionData.get("Runx2")));
    else{
        System.out.println("\
            nstableExpressionData of Dlx5 is "
            +(stableExpressionData.get("Dlx5"))
            +" and should be true in order to
            fit");
        System.out.println("\
            nstableExpressionData of Msx2 is "
            +(stableExpressionData.get("Msx2"))
            +" and should be true in order to
            fit");
        System.out.println("\
            nstableExpressionData of Runx2 is "
            +(stableExpressionData.get("Runx2")
            )+" and should be true in order to
            fit");
        return ((stableExpressionData.get("
            Dlx5")) && (stableExpressionData.
            get("Msx2")) && (
            stableExpressionData.get("Runx2")))
            ;
    }
}
//if the medium under consideration contains
    only TFGb1
else if(TGFb1){
    //model fits the data in the TGFb1
        environment if Runx2 is expressed and
        both Dlx5 and Msx2 are not
```

```
            if(!DEBUGFLAG) return (!(
               stableExpressionData.get("Dlx5")) && !(
               stableExpressionData.get("Msx2")) && (
               stableExpressionData.get("Runx2")));
            else{
                System.out.println("\
                    nstableExpressionData of Dlx5 is "
                    +(stableExpressionData.get("Dlx5"))
                    +" and should be false in order to
                    fit");
                System.out.println("\
                    nstableExpressionData of Msx2 is "
                    +(stableExpressionData.get("Msx2"))
                    +" and should be false in order to
                    fit");
                System.out.println("\
                    nstableExpressionData of Runx2 is "
                    +(stableExpressionData.get("Runx2")
                    )+" and should be true in order to
                    fit");
                return (!(stableExpressionData.get("
                    Dlx5")) && !(stableExpressionData.
                    get("Msx2")) && (
                    stableExpressionData.get("Runx2")))
                    ;
            }
        }
        //This case should not arise (since no media
            is not considered in Kirkham), and thus an
            error is thrown
        else throw new IllegalArgumentException("Error
            !! The case of no GF in the media is not
            covered and should not arise!!");
    }

    /**
     * function to fill the matrix describing the BM
        encoded by index (see documentation for index
        encoding).
     *
```

```
 * @param index The index of the BM, as encoded as
     described in the documentation
 * @return A HashMap with the TFs as keys and an
    array of influence of the relevant species on
    the TF (-1: inhibitory influence, 0: no
    influence, 1: activatory influence). See
    function body or documentation for semantics
    of the data.
 */
private static HashMap<String, int[]> fillMatrix(
   int index){
    /**
     * Matrix to keep the influence of species for
         the three species of investigation (key)
     * Structure of entries is:
     * 0. entry: Influence of BMP2
     * 1. entry: Influence of TGFb1
     * 2. entry: Influence of Dlx5
     * 3. entry: Influence of Msx2
     * 4. entry: Influence of Runx2
     */
    HashMap<String, int[]> matrix = new HashMap
       <>(3);

    //integers to describe the (yet unclear)
       influence of the species unto one another
       as encoded in the index
    int Msx2toDlx5 = ((((int) Math.floor(index /
       3)) % 3) - 1);
    int Runx2toDlx5 = ((int) Math.floor(index % 3)
       - 1);
    int Dlx5toMsx2 = (((int) Math.floor(index /
       81) % 3) - 1);
    int Msx2toRunx2 = (((int) Math.floor(index /
       27) % 3) - 1);
    int TGFb1toRunx2 = (((int) Math.floor(index /
       9) % 3) - 1);

    //entry for the influence of species on Dlx5.
```

```
        //BMP2 has an activatory influence on Dlx5,
            TGFb1 has none, and the influence of Msx2
            and Runx2 on Dlx5 is unclear
        int[] Dlx5Array = {1, 0, 0, Msx2toDlx5,
            Runx2toDlx5};
        matrix.put("Dlx5", Dlx5Array);

        //entry for the influence of species on Msx2.
        //BMP2 and TGFb1 have an activatory influence
            on Msx2, Runx2 has none, and the influence
            of Dlx5 on Msx2 is unclear
        int[] Msx2Array = {1, 1, Dlx5toMsx2, 0, 0};
        matrix.put("Msx2", Msx2Array);

        //entry for the influence of species on Runx2.
        //Dlx5 has an activatory influence on Runx2,
            BMP2 has none, and the influence of Msx2
            and TGFb1 on Runx2 is unclear
        int[] Runx2Array = {0, TGFb1toRunx2, 1,
            Msx2toRunx2, 0};
        matrix.put("Runx2", Runx2Array);

        return matrix;
    }

    /**
     * function to find the steady-state expression
         profile of the BM described by the matrix in a
          medium potentially containing BMP2 and TGFb1.
     * Function will iterate until it finds a stable
         expression profile or iterates too much (more
         than MAXITERATION times), indicating an error
     *
     * @param BMP2 integer indicating whether BMP2 is
         present in the medium (value 1: present, value
          0: not present)
     * @param TGFb1 integer indicating whether TGFb1
         is present in the medium (value 1: present,
         value 0: not present)
```

```
     * @param matrix Matrix describing the BM of
         interest
     * @return A stable expression profile indicating
         whether a TF is expressed in the steady-state
     */
    private static HashMap<String, Boolean>
        stableExpressionProfile(int BMP2, int TGFb1,
        HashMap<String, int[]> matrix){
        //counter to prevent too long oscillations
        int iterationCounter = 0;
        //The initial expression matrix, where none of
            the TFs are expressed
        HashMap<String, Boolean> TFexpressionOld =
            setupExpressionMatrix();
        //The first iteration of the BM described by
            the matrix
        HashMap<String, Boolean> TFexpressionNew =
            updateTFExpression(TFexpressionOld, BMP2,
            TGFb1, matrix);
        //As long as the TF expression differs from
            the previous one (no steady-state reached)
            ...
        while(TFExpressionDifferent(TFexpressionOld,
            TFexpressionNew) && (iterationCounter <
            MAXITERATIONS)){
            TFexpressionOld = TFexpressionNew;
            //...iterate the discrete BM
            TFexpressionNew = updateTFExpression(
                TFexpressionOld, BMP2, TGFb1, matrix);
            iterationCounter++;
            if(DEBUGFLAG){
                System.out.println("\n
                    ----------------------\
                    nTFExpression old is ");
                printTFExpression(TFexpressionOld);
                System.out.println("\n\nTFExpression
                    new is ");
                printTFExpression(TFexpressionNew);
                System.out.println("Difference between
                    them? "+TFExpressionDifferent(
```

```
                TFexpressionOld, TFexpressionNew));
            }
        }
        //indicate that no steady-state has been
            reached
        if(iterationCounter == MAXITERATIONS){
            System.out.println("\n
                ----------------------------------\n");
            System.out.println("Maxiterations reached
                with matrix ");
            printMatrix(matrix);
            System.out.println(" with BMP2 "+BMP2+"
                and TGFb1 "+TGFb1);
            numIndexMaxIt++;
        }
        //expression profile corresponding to a steady
            state
        return TFexpressionNew;
}

private static void printTFExpression(HashMap<
    String, Boolean> tFexpression) {
     for(String tf : tFexpression.keySet()){
            System.out.println("\nExpression for "+tf+
                "\t is "+tFexpression.get(tf));
     }
}

/**
 * function to update the TF expression (i.e. to
     calculate the next time step of the BM under
     interest)
 *
 * @param tFexpressionOld the expression of the
     TFs in the previous step
 * @param BMP2 integer indicating whether BMP2 is
     present in the environment (1) or not (0)
 * @param TGFb1 integer indicating whether TGFb1
     is present in the environment (1) or not (0)
```

```
 * @param matrix the matrix describing the
    influence of the relevant species onto the TF
 * @return a HashMap of the expression values of
    the TFs
 */
private static HashMap<String,Boolean>
   updateTFExpression(HashMap<String, Boolean>
   tFexpressionOld, int BMP2, int TGFb1, HashMap<
   String, int[]> matrix) {
    HashMap<String, Boolean> tFexpressionNew = new
        HashMap<>(3);
    //differentiate whether the influence of the
       species in the BM are connected via
       conjunctions or disjunctions
    switch(BOOLMODE){
        case "AND":
            //in the case of conjunctions,
               calculate the new value of the
               expression of the TFs via
               calculateNewExpressionValueAnd with
                the respective entry in the
               corresponding matrix
            tFexpressionNew.put("Dlx5",
               calculateNewExpressionValueAnd(
               tFexpressionOld, BMP2, TGFb1,
               matrix.get("Dlx5")));
            tFexpressionNew.put("Msx2",
               calculateNewExpressionValueAnd(
               tFexpressionOld, BMP2, TGFb1,
               matrix.get("Msx2")));
            tFexpressionNew.put("Runx2",
               calculateNewExpressionValueAnd(
               tFexpressionOld, BMP2, TGFb1,
               matrix.get("Runx2")));
            break;
        case "OR":
            //in the case of conjunctions,
               calculate the new value of the
               expression of the TFs via
               calculateNewExpressionValueOr with
```

```
                the respective entry in the
                corresponding matrix
            tFexpressionNew.put("Dlx5",
                calculateNewExpressionValueOr(
                tFexpressionOld, BMP2, TGFb1,
                matrix.get("Dlx5")));
            tFexpressionNew.put("Msx2",
                calculateNewExpressionValueOr(
                tFexpressionOld, BMP2, TGFb1,
                matrix.get("Msx2")));
            tFexpressionNew.put("Runx2",
                calculateNewExpressionValueOr(
                tFexpressionOld, BMP2, TGFb1,
                matrix.get("Runx2")));
            break;
        default:
            System.out.println("ERROR!!! BOOLMODE
                SET INCORRECTLY TO "+BOOLMODE);
    }
    return tFexpressionNew;
}

/**
 * calculates the new expression value of the
    species of interest via a disjunction of the
    factors involved
 *
 * @param tFexpressionOld the expression data of
    the current time step (the new time step is
    based upon)
 * @param BMP2 integer to indicate whether BMP2 is
     present in the medium (1) or not (0)
 * @param TGFb1 integer to indicate whether TGFb1
    is present in the medium (1) or not (0)
 * @param matrixRow the influences of other
    species on the TF under consideration as in
    the BM under consideration
 * @return true if the TF is set to 'expressed' in
     the next time step or not
 */
```

```
private static Boolean
    calculateNewExpressionValueOr(HashMap<String,
    Boolean> tFexpressionOld, int BMP2, int TGFb1,
    int[] matrixRow) {
     //if BMP2 is present in the medium and it
         upregulates the TF, the expression value is
          1
     if((BMP2 == 1 ) && (matrixRow[0] == 1)) return
          true;
     //if TGFb1 is present in the medium and it
         upregulates the TF, the expression value is
          1
     else if((TGFb1 == 1 ) && (matrixRow[1] == 1))
         return true;
     //if Dlx5 is expressed and it upregulates the
         TF, the expression value is 1
     else if(tFexpressionOld.get("Dlx5") && (
         matrixRow[2] == 1)) return true;
     //if Dlx5 is not expressed and it inhibits the
          TF, the expression value is 1
     else if(!tFexpressionOld.get("Dlx5") && (
         matrixRow[2] == -1)) return true;
     //if Msx2 is expressed and it upregulates the
         TF, the expression value is 1
     else if(tFexpressionOld.get("Msx2") && (
         matrixRow[3] == 1)) return true;
     //if Msx2 is not expressed and it inhibits the
          TF, the expression value is 1
     else if(!tFexpressionOld.get("Msx2") && (
         matrixRow[3] == -1)) return true;
     //if Runx2 is expressed and it upregulates the
          TF, the expression value is 1
     else if(tFexpressionOld.get("Runx2") && (
         matrixRow[4] == 1)) return true;
     //if Runx2 is not expressed and it inhibits
         the TF, the expression value is 1
     else if(!tFexpressionOld.get("Runx2") && (
         matrixRow[4] == -1)) return true;
     else return false;
}
```

```
/**
 * calculates the new expression value of the
    species of interest via a conjunction of the
    factors involved
 *
 * @param tFexpressionOld the expression data of
    the current time step (the new time step is
    based upon)
 * @param BMP2 integer to indicate whether BMP2 is
     present in the medium (1) or not (0)
 * @param TGFb1 integer to indicate whether TGFb1
    is present in the medium (1) or not (0)
 * @param matrixRow the influences of other
    species on the TF under consideration as in
    the BM under consideration
 * @return true if the TF is set to 'expressed' in
     the next time step or not
 */
private static Boolean
   calculateNewExpressionValueAnd(HashMap<String,
   Boolean> tFexpressionOld, int BMP2, int TGFb1,
   int[] matrixRow) {
    //if BMP2 upregulates the TF and is not
       present in the medium, the expression value
        is 0
    if((BMP2 == 0 ) && (matrixRow[0] == 1)) return
        false;
    //if TGFb1 upregulates the TF and is not
       present in the medium, the expression value
        is 0
    else if((TGFb1 == 0 ) && (matrixRow[1] == 1))
       return false;
    //if Dlx5 upregulates the TF and is not
       expressed, the expression value is 0
    else if(!tFexpressionOld.get("Dlx5") && (
       matrixRow[2] == 1)) return false;
    //if Dlx5 inhibits the TF and is expressed
       expressed, the expression value is 0
```

```
        else if(tFexpressionOld.get("Dlx5") && (
            matrixRow[2] == -1)) return false;
        //if Msx2 upregulates the TF and is not
            expressed, the expression value is 0
        else if(!tFexpressionOld.get("Msx2") && (
            matrixRow[3] == 1)) return false;
        //if Msx2 inhibits the TF and is expressed,
            the expression value is 0
        else if(tFexpressionOld.get("Msx2") && (
            matrixRow[3] == -1)) return false;
        //if Runx2 upregulates the TF and is not
            expressed, the expression value is 0
        else if(!tFexpressionOld.get("Runx2") && (
            matrixRow[4] == 1)) return false;
        //if Runx2 inhibits the TF and is expressed,
            the expression value is 0
        else if(tFexpressionOld.get("Runx2") && (
            matrixRow[4] == -1)) return false;
        else return true;
    }


}
```

Appendix C: Results fixed point analysis

This appendix will list the results of the fixed point analysis of the boolean models identified as stable by [2].

As noted in 3.1.2, some assumptions were not explicitly stated in the aforementioned publication. In order to identify the assumptions their approach is based upon, a systematic investigation of these assumptions is performed in the following.

The three models identified by [2] are coded as the cases, where the influence of Runx2 on Dlx5 will be activatory (case 1), inhibitory (case 2) or no influence of Runx2 on Dlx5 will be assumed (case 3). Each case itself will contain three subcase specifying the environment (x.1: only BMP2 present, x.2: only TGFβ1 present, x.3: BMP2 as well as TGFβ1 present).

Furthermore are the cases distinguished by whether the logical connection in the boolean models is conjunctive or disjunctive.

An additional assumption tested is whether in the absence of a growth factor (BMP2 / TGFβ1), a constant 'Off' is assumed for this variable in the boolean model, or whether this variable is ignored.

The following 18 cases will assume that in the absence of a growth factor the variable will be considered *Off*.

case 1.1: Runx2 activatory on Dlx5, BMP2 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	Off	Off

→ A fixed point is reached no species being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 1.2: Runx2 activatory on Dlx5, TGFβ1 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	Off	Off

→ A fixed point is reached with no species being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	On
2	On	On	Off
3	On	On	On

→ A fixed point is reached with all species being expressed

case 1.3: Runx2 activatory on Dlx5, BMP2 and TGFβ1 present:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	Off	On	Off

→ A fixed point is reached with Msx2 being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 2.1: Runx2 inhibitory on Dlx5, BMP2 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	Off	Off

→ A fixed point is reached no species being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 2.2: Runx2 inhibitory on Dlx5, TGFβ1 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	Off	Off

→ A fixed point is reached with no species being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 2.3: Runx2 inhibitory on Dlx5, BMP2 and TGFβ1 present:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	Off

→ A cycle is observed where Dlx5 and Msx2 switch each other on and off (mutually)

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 3.1: no influence of Runx2 on Dlx5, BMP2 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	Off	Off

→ A fixed point is reached no species being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 3.2: No influence of Runx2 on Dlx5, TGFβ1 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	Off	Off

→ A fixed point is reached with no species being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 3.3: No influence of Runx2 on Dlx5, BMP2 and TGFβ1 present:

conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	Off

→ A cycle is observed where Dlx5 and Msx2 switch each other on and off (mutual, alternating, no activity)

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

In the following cases, the absence of a GF in the medium will be interpreted as the absence of the factor in the boolean model. The results of the fixed point analysis under this assumption is presented in the following:

case 1.1: Runx2 activatory on Dlx5, BMP2 only:

conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	Off	On	Off

→ A fixed point is reached with Msx2 being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 1.2: Runx2 activatory on Dlx5, TGFβ1 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	Off	On	Off

→ A fixed point is reached with Msx2 being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	On
2	On	On	Off
3	On	On	On
4	On	On	On

→ A fixed point is reached with all species being expressed

case 1.3: Runx2 activatory on Dlx5, BMP2 and TGFβ1 present:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	Off	On	Off

→ A fixed point is reached with Msx2 being expressed.

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 2.1: Runx2 inhibitory on Dlx5, BMP2 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	Off

→ An oscillatory cycle with the behaviour sketched above (alternating activation and mutual inhibition of Dlx5 and Msx2) is observed

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 2.2: Runx2 inhibitory on Dlx5, TGFβ1 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	Off

→ An oscillatory cycle with the behaviour sketched above (alternating activation and mutual inhibition of Dlx5 and Msx2) is observed

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 2.3: Runx2 inhibitory on Dlx5, BMP2 and TGFβ1 present:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	Off

→ An oscillatory cycle with the behaviour sketched above (alternating activation and mutual inhibition of Dlx5 and Msx2) is observed

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 3.1: no influence of Runx2 on Dlx5, BMP2 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	On
5	Off	On	Off

→ An oscillatory cycle with the behaviour sketched above (alternating activation and mutual inhibition of Dlx5 and Msx2 and occasional activation of Runx2 when Dlx5 is expressed and Msx2 is not expressed) is observed

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

case 3.2: No influence of Runx2 on Dlx5, TGFβ1 only:
conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	Off

→ An oscillatory cycle with the behaviour sketched above (alternating activation and mutual inhibition of Dlx5 and Msx2) is observed

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	On
2	On	On	Off
3	On	On	On
4	On	On	On

→ A fixed point is reached with all species being expressed

case 3.3: No influence of Runx2 on Dlx5, BMP2 and TGFβ1 present:

conjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	Off	On	Off
2	On	On	Off
3	On	Off	Off
4	Off	Off	Off

→ An oscillatory cycle with the behaviour sketched above (alternating activation and mutual inhibition of Dlx5 and Msx2) is observed

disjunction:

t	Dlx5	Msx2	Runx2
0	Off	Off	Off
1	On	On	On
2	On	On	On

→ A fixed point is reached with all species being expressed

Appendix D: Odefy simulation of possible candidate networks

To model the relations under investigation not evident from the literature, 5 variables are used, where the value -1 represents an inhibitory relation, 0 no influence of the first species onto the second and 1 as activatory influence. The variables used are as follows:

variable	biological meaning	value
Msx2Dlx5	Influence of Msx2 on Dlx5	(index / 3^0) % 3 - 1
Dlx5Msx2	Influence of Dlx5 on Msx2	(index / 3^1) % 3 - 1
Msx2Runx2	Influence of Msx2 on Runx2	(index / 3^2) % 3 - 1
TGFb1Runx2	Influence of TFGβ1 on Runx2	(index / 3^3) % 3 - 1
Runx2Dlx5	Influence of Runx2 on Dlx5	(index / 3^4) % 3 - 1

Model creation

The BMs specified above are 'odified' through the *ExpressionsToOdefy* command of the odefy toolbox. This command requires an expression describing the boolean network. For every candidate configuration of the GRN the interactions of the species are modeled. For all three genes, the relevant interactions are either added as activatory (x), left out or added as inhibitory ($\tilde{x}$).

For this project, default parameters were chosen [8] , since no parameters were reported in [2], and it is to be expected that default parameters were used in the publication. The following source code exemplifies how the expressions are constructed:

```
models = [0:242]
for index = [0 : 242]

# behavior of Msx2
Msx2String = "Msx2 = TGFb1 \&\& BMP2";
if(rem(floor(index / 3), 3) - 1 == -1)
        Msx2String = [Msx2String, " \&\& ~Dlx5"];
elseif(rem(floor(index / 3), 3) - 1 == 1)
        Msx2String = [Max2String, " \&\& Dlx5"];
endif
```

[8] n=3, k=0.5, τ=1, according to [3]

```
# behavior of Dlx5
Dlx5String = "Dlx5 = BMP2"
#       influence of Msx2
if(rem(floor(index), 3) - 1 == -1)
        Dlx5String = [Dlx5String, " \&\& ~Msx2"]
elseif(rem(floor(index), 3) - 1 == 1)
        Dlx5String = [Dlx5String, " \&\& Msx2"]
endif
#       influence of Runx2
if(rem(floor(index / 81), 3) - 1 == -1)
        Dlx5String = [Dlx5String, " \&\& ~Runx2"]
elseif(rem(floor(index / 81), 3) - 1 == 1)
        Dlx5String = [Dlx5String, " \&\& Runx2"]
endif

# behavior of Runx2
Runx2String = "Runx2 = Dlx5"
#       influence of Msx2
if(rem(floor(index / 9), 3) - 1 == -1)
        Runx2String = [Runx2String " \&\& ~Msx2"]
elseif(rem(floor(index / 9), 3) - 1 == 1)
        Runx2String = [Runx2String " \&\& Msx2"]
endif
#       influence of TGFb1
if(rem(floor(index / 27), 3) - 1 == -1)
        Runx2String = [Runx2String " \&\& ~TGFb1"]
elseif(rem(floor(index / 27), 3) - 1 == 1)
        Runx2String = [Runx2String " \&\& TGFb1"]
endif

# construct BM for odefy
expression = {"BMP2 = <>", " TGFb1 = <>",Msx2String,
   Dlx5String, Runx2String};
model = ExpressionsToOdefy(expression);
models(index) = model
endfor
```

As mentioned in [3], the boolean update functions are accessible as multidimensional arrays (hypercubes of edge length two). These are accessible through

```
model.tables(x).inspecies
```

for a list of species that influence the species, and

```
model.tables(x).truth
```

for the truth table for the expression the model was based upon. In the above code snippets, x stands for the index of the table struct to be accessed. Using *ExpressionToOdefy*, the full BM is stored as an Odefy-internal representation with Octave.

Analogue to this, expressions can be constructed employing disjunctions instead of conjunctions for the expressions of the boolean models. These (continuous) models can subsequently be analyzed as described in 3.2.3. Due to the large number of these models and the extend of this course work however, this will not be elaborated further, and is left for the interested reader.

Simulation of continuous models

After the construction of the BM sketched above, simulations of the networks are performed using the *OdefySimulation* command. This command takes a simstruct as a parameter which has to be constructed from the models using the *CreateSimstruct* command. The code thus is as follows:

```
# construct struct for simulation
simstruct = CreateSimstruct(model);
simstruct.type = 'hillcubenorm';
# set presence of BMP2
simstruct = SetInitialValue(simstruct,'BMP2',1);
# set presence of TGFb1
simstruct = SetInitialValue(simstruct,'TGFb1',0);
# simulate
simres = OdefySimulation(simstruct);
plot(simres);
legend(model.species);
xlabel('time');
ylabel('expression')
```

The *simstruct* is a data structure that holds the information required for the simulation, like initial states, simulation type and parameters.

In the example above, the initial values of the GFs are set arbitrarily to 'on' (BMP2) and 'off' (TGFβ1) to exemplify the simulation run in the *BMP2 only* environment. This is set purely for illustrative purposes, and all three

setups of the environment presented in [2] are employed in the simulation. The *type* attribute of the simstruct specifies that the normalized HillCube variant is employed.

The simulation is invoked by the *OdefySimulation* function that takes the simstruct as parameter. Consequently, the expression of the species used is plotted.

References

[1] D.M. Wittmann et. al. Transforming boolean models to continuous models: methodology and application to t-cell receptor signaling. *BMC Systems Biology*, 3(98), 2009.

[2] G.R. Kirkham et. al. Early gene regulation of osteogenesis in embronic stem cells. *Integrative biology*, 4(12):1470–7, 2012.

[3] J. Krumsiek et. al. From discrete to continuous gene regulation models - a tutorial using the odefy toolbox. *Applications of MATLAB in Science and Engineering*, 2011.