Boolean network modelling practical 14-10-2021

In this practical, we will start by looking at a Boolean network model for the plant root stem cell
niche (SCN). The paper by Azpeita et al. (2010) from which we have taken this
model investigates how it is possible that this niche, though it contains stem cells that grow and
whose progeny differentiates throughout the plant's life, nevertheless has a very stable spatial ordering (see Fig 2 of the paper).

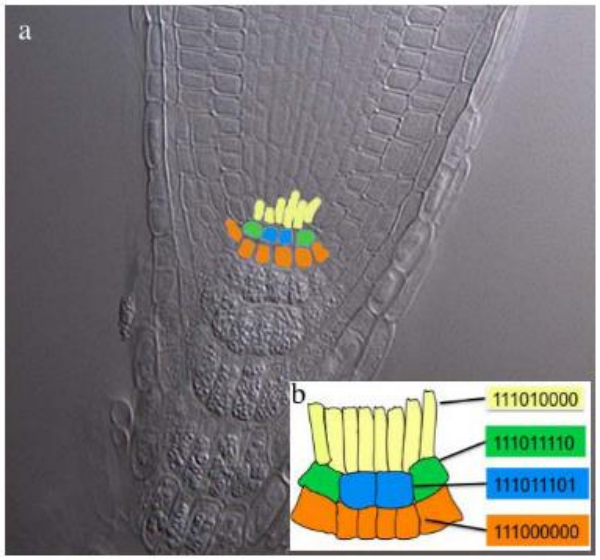
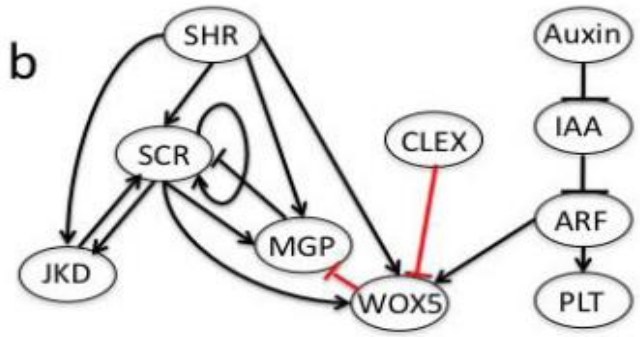


Figure 2 Simplified cellular pattern of the root stem cell niche compared to a real root. The four cell types within the SCN, each of which corresponds to one of four stable attractors recovered by the models in figure 1. The vascular initials (yellow), cortex-endodermis initials (green), quiescent center (blue) and columella-epidermis-lateral root cap initials (orange) (b) in the cleared root tip of *Arabidopsis thaliana* colorized to show the corresponding cell types represented by the attractors recovered by the single-cell GRN models in figure 1 and schematized in (a). The activation states of the nodes are represented by *0* and *1* in the following order: *PLT*, auxin, *ARF*, *Aux/IAA*, *SHR*, *SCR*, *JKD*, *MGP* and *WOX5*.

In the paper, the authors used experimentally known interactions between genes in the plant root to
explain the existence of 4 distinct cell types (of which one is actually composed of 2 subtypes that
the authors lumped together). In their investigations they tried out 3 different model variants. For
this practical we will focus on model B (Fig 1B), which consists of 10 genes, including one that is
unidentified but that the authors dubbed CLEX as it might be a CLE4-like factor. In table 3 of the
paper the authors list the attractors of this model and their correspondence to the root tip cell types.



This practical aims to give you some practice with working with boolean network models. The chosen model is still of relatively limited complexity compared to e.g. the Toll receptor network illustrated in the lecture. The advantage is that it enables us to study it exhaustively, and
keep track of which gene is which.

Before you start on the questions, have a look at the paper. Read the abstract, look at figure 1b, 2 and table 3 to familiarise yourself with some terms. Then, open BooleanNetAssignmentFinalStudentVersion.R and walk through it, completing the questions. Good luck!

Question 1
Load the model "MODEL_Azpeitaetal.xml" and visualise it. Be aware that the plotNetworkWiring function sometimes plots self-interactions (feedback of a gene on itself) through other nodes, which can look confusing. How many states does this network have?

A: Given that there are 10 genes, which each have two possible states the total number of states =2^10=1024

Question 2
What are the largest attractors of this network?

A:Attractor 1 (only AUX/IAA on) and 2 (Auxin, ARF, PLT on) are the largest, with a basin of attraction of 256 states

Question 3
Attractor 1 has a large basin of attraction and simply consists of AUX/IAA being on
and all other genes being off. Looking at the network architecture, can you describe why this is such
a common end state if we model the system from all possible starting points? Can you explain why
this happens in exactly 256 states?

*A: If AUXIN is off (which will be the case in half of the possible 1024 states, so 512 states), independent of initial conditions AUX/IAA will turn on, ARF will turn off and hence nothing can be
activated by ARF. If additionally SHR is off (which will be the case in half of the 1024 states, so 512
states) all the genes that depend on it for activation will go off as well, independent of their initial*

condition. To go to a situation with only AUX/IAA on, we thus need to start from any situation with
AUXIN off and SHR off. Fixing 2 of the 10 genes to a particular state, leaves 2^8=256 possible states for the remaining genes. This particular attractor is not biologically relevant for the stem cell
niche as auxin levels will always be high there.

Question 4
Test your answer from the question above by fixating the two genes to only obtain this single attractor. Use the fixGenes() function for this together with the original network.

A: #R code
```
fixedNetAuxinSHRZero <- fixGenes(netReorder, fixIndices = funcSetNames(netReorder, c("AUXIN", "SHR")),
                values = c(0, 0))
newAttractorsKnockoutAuxinAndSCR <- getAttractors(fixedNetAuxinSHRZero)
print(newAttractorsKnockoutAuxinAndSCR)
```

Question 5
Can you now also explain why attractor 2 is so common?

A: If Auxin is on (which is in half of the 1024 so in 512 cases), independent of initial conditions AUX/IAA will turn of, ARF will turn on and PLT will turn on. To ensure that no other genes are on, SHR needs to start in the off state (see previous question). Thus to end up in this attractor, Auxin needs to start in on state and SHR in off state, so again we fix the state of 2 of the 10 genes, leaving
2^8=256 possible states to start from. Note that this attractor corresponds to the epidermis-lateral
root cap initials (CepI) from the paper.

Question 6
You can also plot the entire state space with all the attractors. Here, you plot every state of the system as a node, and visualise the transitions between them with arrows. Attractors are
the final states that the whole network ends up in. Visualise these attractors now. Be sure to use the
zoom function of your R plot window. What do you notice about the basin structure of attractor 1
and attractor 2?

A: The basins of attraction of these attractors look extremely similar. This is unsurprising, given that they both are attractors for which the state of Auxin and SHR had to be fixed in order for an initial state to end up there

Question 7
Find which of the other attractors we found correspond to which attractors discussed in the paper
(table 3). What is the basin of attraction (size) of these 3 other paper attractors?

A:

Quiescent center (QC) = attractor 6 = 48 states
Vascular initials: 2 attractors, one with CLEX on, one with CLEX off:
Attractor 8, 80 states, CLEX on
Attractor 4, 80 states, CLEX off. → Combined thus 160 states that lead to it.
Epidermis lateral root cap (CepI) = attractor 2 = 256 states.
CEI = attractor 10 = 48 states
Note that we thus have 5 attractors that aren't mentioned in this paper at all (of which you already know that attractor 1 is biologically not very relevant)!

Question 8
You now know which attractors correspond to those in the paper, and that attractor 1 is not so interesting. There are other attractors. Let's zoom in on attractor 5, what does that correspond to, and what does it tell you about the promoter of WOX5 (i.e. how it is activated)? Check this in
the rules.

A: 5: AUX/IAA SHR SCR JKD MGP on, others off. Basin of attraction 48.
From the figure of the network model, you might think that if SHR and SCR are on, and CLEX is off, WOX5 could be active: it has an activating connection from both SHR and SCR, and its main repressor is absent. Nevertheless, it is off. In the rules, you see that we assume that WOX5 is only active when SCR & ARF & SHR & !CLEX, i.e. it is only on when all three activators, so also ARF, are binding its promotor and no repression by CLEX is taking place. It must have quite a complex promotor!

Question 9
So far we have applied the updating rules at the same time for all nodes, called synchronous updating. However, transcription, translation and protein degradation do not necessarily occur at the
same rate for all genes, and additionally these processes are inherently noisy. Thus, synchronous
updating of all genes may not be realistic. Let us instead consider asynchronous updating, were we
update every node, one at a time, in a random order until we have updated every node (and repeat
this until the network converges on some attractor). We will first show the difference on an example
network of the mammalian cell cycle. Load it by typing *data("cellcycle")* Plot this network, and first look at the attractors when updating synchronously as we did before (i.e. use *plotStateGraph()*). What do you see?

A: There are two attractors, one is cyclic,one is a single-state attractor.

Question 10
Which attractor might correspond to an actively dividing cell, and which to a quiescent cell?

A: Attractor 1, with its cyclic attractor, is indeed cycling through cell states and will probably correspond to actively dividing cells. Attractor 2, in contrast, will not cycle unless this state is perturbed by some signalling to go into an active cell cycle again. Hence, it seems likely that this is
some quiescent state.

Question 11
Now, let's update the network asynchronously (use type=*'asynchronous'*, *method = 'random'*, and
startStates = 800, for example) and plot the attractors you get. What happens? Why would this happen?

A: The well-specified cyclic attractor is gone. A single-state attractor remains, together with a complex loose attractor with 112 states and 338 transitions. What does that mean? Well, it is a non-unique cycle or loop of states that are all connected. It looks like Figure 1B below:

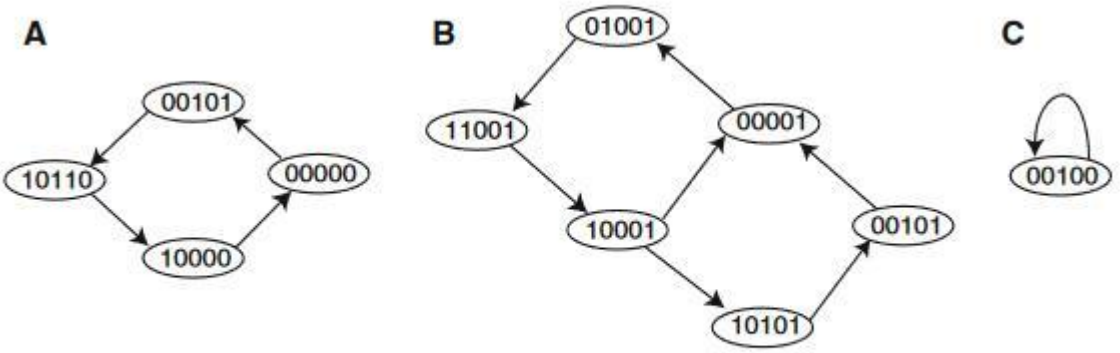


Fig. 1 **a** A simple cycle in a synchronous Boolean network. **b** A complex attractor in an asynchronous Boolean network. **C** A steady-state attractor in a synchronous or asynchronous Boolean network

(ref: Hopfensitz, M., Müssel, C., Maucher, M., & Kestler, H. A. (2013). Attractors in Boolean networks: a tutorial. Computational Statistics, 28(1), 19-36.). It is a grid of states, all connected, but because of asynchronous updating it is non-unique: in 1B above you might go from 10001 to 00001 or to 10101, depending on what gene is randomly chosen to update in this time step. You can imagine that this is quite a bit less organised then a nice, robust (cell) cycle like in figure 1A above: depending on which node is updated at what time, you might skip steps completely, or get a longer cycle. This is not good for something as rigidly controlled as a cell cycle. Hence, this cycle is not resistant to changing to the asynchronous update regime. Why are we interested in trying this in the first place?

The meta point is that all models are caricatures in a way. Synchronous updating is a caricature because it assumes that there is a universal tick of the clock at which all genes update their state all at the same time. That's quite preposterous: things are random! There is diffusion and Brownian motion, certain genes are made at different levels or speeds than others (because of length, posttranslational modification, etc.). Another extreme is asynchronous updating: rather than letting all genes update all at once, you say that at every tick of some clock only 1 gene updates, and then another, and then another, etc., until you've had all 10. Then you again take a random order and update them one-by-one, etc. This is perhaps also preposterous, but preposterous in a different way. If you see similar behaviours in different caricatures (different model formalisms with different assumptions), then you are more sure that the biological behaviours you observe are similar to reality, rather than just flukes due to your specific choice of updating rule (synchronous or asynchronous) or model formalism (FBA or ODE or Boolean Network models, or...)

The reason all this happens is because where before in synchronous updating you might go from 00110 → 01001, now you have 5 options depending on which gene you update first. So 00110 → 10110 ; 00110 → 00110 (2nd gene from the left still 0) ; 00110 → 00010; 00110 → 00110 (4th gene from the left stays 1) ; 00110 → 00111. So there are many more options when updating and this influences the behaviour of the system.

Question 12
We go back to our original root tip model. Get asynchronous attractors from the root SCN network.
What do you observe? (Note: if you use *plotAttractors()* or when you *print()* the attractors, the order
that the attractors appear in will be different).

A: Nothing changes whatsoever. This is not so strange upon reflection: 4/10 genes are not doing much in the network (Auxin, IAA, ARF, PLT). For the other 6, SHR is the main activator of 5/6 of them (also of CLEX, although not shown in the figure). There were already no cycles, only single state attractors, so it makes sense that even when updating genes in a random order, not that much
will change! It is hence robust to a change in timing regime of the model. In fact, in the paper, the authors also use an ODE-like model with concentrations of genes and continuous time, and the model still works well.

Question 13
If you fix SHORTROOT at 0 (loss of function mutation), do you expect that
SCARECROW (SCR) could still keep part of the network activated? Try it (Go back to synchronous updating to be safe) What happens, and can you check the rules to see why this is?

A: From the network picture, you might imagine that SCR could sort of take over for SHR: it also has activating connections to JKD and MGP and activates itself. However, the model rules all stipulate that both SHR and SCR should be on for these genes, so no it can not take over.

Question 14
We might wonder how the results change if we change the rules of the network, rather
than the states of the nodes (i.e. mutate regulatory interactions rather than genes). For this, we can
use the function *perturbNetwork()*, which changes one of the regulatory interactions. Be sure to set
perturb = "functions", method = "bitflip", readableFunctions = TRUE inside the function. Run it once on the initial network. What has changed? Do you see the change when plotting or in the Boolean functions? Do this two or three more times and see what changes.

To get a better idea of what is happening here:
Assume genes A and B control the expression of gene C such that C=(A AND B), so in other words,
C=1 if A=1 and B=1 and 0 otherwise. Another way of writing this is as follows: First, gene C has 2
input genes, so there is a total of 2^2=4 possible input states (AB: 00, 01, 10, 11), let us order these
according to the numeric value of their binary code, since 00=0, 01=1, 01=2, 11=3, this will be 00;
01; 10; 11. Now, let us write down how gene C responds to each of these input states: only A and B
on gives C=1 so 00 → 0, 01 → 0, 10 → 0, 11 → 1. The update rule for gene C can now be written by only writing the ordered output for gene C: 0001. To see for our network what
an update rule for e.g. SCR looks like you can use *YOURNETWORKNAME$interactions$SCR*, and

look what is listed under func. What the perturbNetwork() function does is flip a bit in this update
rule. If eg. 0001 changes into 0000 in the above example it means that gene C is switched on by
none of the possible inputs it receives and hence is effectively knocked out. Note that to interpret
the binary update rules you need to know which genes affect the gene, and in which order.

A. Depending on the changes made, either network architecture only, or also attractors and basins of attraction may change.

Question 15: Although Boolean network models are also suitable for simulating large networks this is not entirely trivial. Consider a network with 100 nodes. How many possible states does this network have? What consequences does this have?

A: A whopping 2^100, so it becomes impossible to simply try out all states to see what attractors they converge to. Indeed, subsampling and tailored algorithms are now necessary to analyze the behavior of the network. In fact, the number of atoms in the observable universe is about 10^80. How many nodes do you think it takes to get that many states? Turns out, it's only ~266 genes. So even with the simplification into 1 or 0 as states of a gene if you simulate a network with 266 genes you get more possible states than there are atoms in the observeable universe.
Calculation:
10^80 = 2^x
ln(10^80) = ln(2^x)
80 ln(10)/ln(2) = x*
x ~ 266

Optional final questions (Really optional. Feel free to stop and celebrate that you're done!)

Question 16

(Feel free to take a peek in the answer script if you don't manage to do this yourself)
Use set.seed(666999) (just to ensure your findings match ours). Make a simple loop that perturbs
the network randomly 70 times. Save the resulting networks to a list, and save the attractors in a
different list. What are the minimum and maximum number of attractors you obtained?

A: The number of attractors is mostly stable at 10. There is one change that caused the number of
attractors to increase to 12. The lowest number of attractors observed is 2.

Question 17

Take a look at the original network as well as your previous answers. Which
regulatory change would cause a change to 2 attractors? And which one to 12? If you can not guess
have a look at the rules of the perturbed networks generating these attractor numbers and compare
them to the original network.

A: Based on the question where we fixed SHR, it would be logical to assume that the rule for SHR has changed from 0 1 (remains inactive if inactive, if active self-activates) to 0 0, meaning that it is
basically knocked out. Then we get only 2 attractors (Auxin on or off).
In the case of the 12 attractors, the rule for CLEX has changed from 0001 to 0101. In other words, CLEX now activates itself (its next state will be 1 if CLEX is 1). CLEX was first limited by SHR and now isn't anymore, so it can be on in two more attractors.