University of Luxembourg

User Guide of CABEAN

Cui Su

January 24, 2021

Contents

1 Introduction

CABEAN is a software tool for the control of large asynchronous Boolean networks, which are often used to model biological systems. The control of a Boolean network aims to identify effective perturbations that can drive the dynamics of the network from the given source attractor or from any initial state to the desired attractor. Currently, CABEAN provides several control methods:

- the minimal one-step instantaneous source-target control (OI) [6, 7];
- the minimal one-step temporary source-target control (OT) [10];
- the minimal one-step permanent source-target control (OP) [10];
- attractor-based sequential instantaneous source-target control (ASI) [3];
- attractor-based sequential temporary source-target control (AST) [9];
- attractor-based sequential permanent source-target control (ASP) [9];
- instantaneous target control (ITC) [8];
- temporary target control (TTC) [8];
- permanent target control (PTC).

Prior to the computation of control, CABEAN performs the identification of attractors using the decomposition-based attractor detection method [4]. This method is able to identify all the exact attractors of the network. Users can then specify the indices of the source and target attractors for control. Currently, the source and target attractors can only be one of the attractors. The attractor can be either a single-state attractor or a cyclic attractor. If the source attractor is cyclic, CABEAN computes the control from a state in the source attractor that requires the fewest number of perturbations. The status of the target attractor, being either a single-state attractor or a cyclic attractor, has no influence on the control results.

2 Installation Guide

2.1 System Requirement

1. Platform: x86 compatible 32 bit or 64 bit processor;
2. Operating system: Windows, Linux and Mac OSX (10.4 and above);
3. Required compilers: flex 2.6.4 or higher, GNU bison 3.0.4 or higher, GNU g++ 4.0.1 or higher, and Windows subsystem for Linux (Windows platform only).

2.2 Installation

The executable binary of CABEAN can be downloaded at *http://satoss.uni.lu/software/CABEAN/*. Please send any feedback to cui.su@uni.lu. With the required compiler, the binary of CABEAN can be executed directly with command line. The syntax of the input file and the command options are universal for different operating systems,

nodes	functions	indices of attractors	attractors
x_1	$f_1 = x_2$	A_1	001100
x_2	$f_2 = x_1$	A_2	001111
x_3	$f_3 = (\sim x_2) \vee x_4$	A_3	110000
x_4	$f_4 = x_3$	A_4	110010
x_5	$f_5 = (x_1 \vee x_6) \wedge x_5$	A_5	111100
x_6	$f_6 = x_4 \wedge x_5$	A_6	111111

Table 1: Boolean functions and attractors of the example.

```
targets, factors
x1, x2
x2, x1
x3, (!x2)|x4
x4, x3
x5, (x1|x6)&x5
x6, x4&x5
```

Figure 1: The model file under BoolNet format.

3 Syntax of the Input Files

CABEAN accepts three types of input files, including the model file, files for the specification of undesired attractors and undesired perturbations.

3.1 Model File

The model file can be under BoolNet or ISPL format. The primary option is ISPL format. If the model file is a BoolNet file, CABEAN will first convert BoolNet to ISPL and then encode the Boolean network into a binary decision diagram (BDD).

Although CABEAN does not support other formats, such as SBML-qual and GINSim, they can be easily converted to BoolNet using BioLQM toolkits [5],[1] which support the conversion of several popular formats for logical qualitative models.

BoolNet Format

The major advantage of BoolNet format [1] lies in its simplicity. BoolNet describes a Boolean network in a standardized text file format. A BoolNet file has a header (first line): "targets, factors". The name and the Boolean function of each node are separated by a comma at each line. The header implies the format of each line: "targets" is the name of the node and "factors" represents the Boolean function of the node. In the Boolean functions, the logical operators 'and', 'or' and 'not' are represented as '&', '|', and '!', respectively. Users can initialise the value of an input node to either '0' or '1' in the BoolNet file.

To give you an example, the Boolean functions and attractors of a toy Boolean network are given in Table 1. The model file under BoolNet format is given in Figure 1.

[1]BioLQM is available at *http://colomoto.org/biolqm/*.

ISPL Format

The ISPL format is the primary option for the model files. We refer details of ISPL format to *https://vas.doc.ic.ac.uk/software/mcmas/manual.pdf*. Here, we use the example given in Table 1 to introduce the basics of ISPL format. The model file of the example under ISPL format is given in Figure 2. It contains two parts: 'Agent M' and 'InitStates'.

1. Section 'Agent M' defines the Boolean variables and their Boolean functions. It can be further split into four subsections:
   (a) subsection 'Vars' defines Boolean variables of the network;
   (b) subsections 'Actions' and 'Protocol' are mandatory but we define them as 'none' as shown in Figure 2;
   (c) subsection 'Evolution' defines Boolean functions of the variables. The logical operators 'and' and 'or' are also represented as '&' and '|'. Different from BoolNet format, the logical operator 'not' is denoted by '$\sim$'.

2. Section 'InitStates' defines the initial state. A prefix "M." is added to each node x to indicate that x is a variable defined in agent M. Note that the initialisation of an input node in "InitStates" will shrink the state space to the states, in which the input node has the initial value. Thus, it will influence the attractors and the control results. However, the initialisation of a non-input node has no influence on the attractors or the control results.

3.2 Specification Files

Encoding a phenotype. CABEAN supports the target control of a target phenotype. The specification file for encoding a phenotype has a header: "node, value". Each line contains the name and the value of a marker node, separated by comma. The value could be either '0' or '1'.
Encoding constraints. CABEAN integrates practical constraints on the attractors and perturbations during the computation. User can define a list of undesired attractors that can not play the role of source, target or intermediate attractors. The indices of undesired attractors are specified in a text file and the indices are separated by comma. Suppose attractors A_2 and A_5 are undesired attractors. The specification file is shown below.

```
2,5
```

User can define three types of undesired perturbations:

- `R0`: nodes that can not be perturbed from 'off' to 'on';
- `R1`: nodes that can not be perturbed from 'on' to 'off';
- `R`: nodes that can not be perturbed in either way.

```
Agent M
   Vars:
      $x_1$:  boolean;
      $x_2$:  boolean;
      $x_3$:  boolean;
      $x_4$:  boolean;
      $x_5$:  boolean;
      $x_6$:  boolean;
   end Vars
   Actions = none;
   Protocol:
      Other:  none;
   end Protocol
   Evolution:
      $x_1$=true  if $x_2$=true;
      $x_1$=false if $x_2$=false;
      $x_2$=true  if $x_1$=true;
      $x_2$=false if $x_1$=false;
      $x_3$=true  if $(\sim x_2)|x_4$=true;
      $x_3$=false if $(\sim x_2)|x_4$=false;
      $x_4$=true  if $x_3$=true;
      $x_4$=false if $x_3$=false;
      $x_5$=true  if $(x_1|x_6)\&x_5$=true;
      $x_5$=false if $(x_1|x_6)\&x_5$=false;
      $x_6$=true  if $x_4\&x_5$=true;
      $x_6$=false if $x_4\&x_5$=false;
   end Evolution
end Agent

InitStates
      $M.x_1$=true or $M.x_1$=false;
end InitStates
```

Figure 2: The model file under ISPL format of the example.

For each type, the nodes are separated by comma as shown below.

```
R0:
R1:  $x_5$
R: $x_4, x_6$
```

4 Execution of CABEAN with Command Line

4.1 Command Options

We summarise the command options as follows.

- `-compositional <number>`: choose the method for attractor detection and basin computation. Only options '0' and '2' are supported in the current version. Option '0' is the default option and it indicates that the global methods for the computation of attractors [4] and basins [6] are selected. Option '2' means the decomposition-based computation of attractors [4] and basins [6] will be used.

- `-control <control method>`: choose one of the control methods:
  - `OI`: the minimal one-step instantaneous source-target control;
  - `OT`: the minimal one-step temporary source-target control;
  - `OP`: the minimal one-step permanent source-target control;
  - `ASI`: the attractor-based sequential instantaneous source-target control;
  - `AST`: the attractor-based sequential temporary source-target control;
  - `ASP`: the attractor-based sequential permanent source-target control.
- `-sin <number>` : specify the index of the source attractor for source-target control.
- `-tin <number>`: specify the index of the target attractor for source-target control and target control.
- `-allpairs`: for source-target control methods, compute results for all combinations of source and target attractors.
- `-tmarker <file name>`: specify a phenotype for target control. '<file name>' denotes the path of the specification file.
- `-rmID <file name>`: set the indices of undesired attractor for source-target control. The cases where the source or the target is undesired will be skipped. The undesired attractors will also be avoided as intermediate attractors for ASI control. Note that '<file name>' is the path of the specification file.
- `-rmPert <file name>`: set undesired perturbations for source-target control. Similarly, '<file name>' represents the path of the specification file.
- `<model file>`: the path of the model file is always placed at the end of the command line. Note that the location of the executable binary is referred as the current directory.

4.2 Examples of Command Lines

In this section, we summarise some general command lines.
Attractor detection. Attractors are computed with the following command line:
```
./cabean -compositional 2 <model file>
```
Source-target control. Below are the commands for source-target control. We use '<control>' to denote one of the control methods (OI, OT, OP, ASI, AST and ASP).

1. Computation of the control for one pair of source and target attractors:
   ```
   ./cabean -compositional 2 -control <control> -sin <index of the source> -tin <index of the target> <model file>
   ```
2. Computation of the control for one pair of source and target attractors with constraints on perturbations:
   ```
   ./cabean -compositional 2 -control <control> -sin <index of the source> -tin <index of the target> -rmPert <file name> <model file>
   ```

3. Computation of the control for one pair of source and target attractors with constraints on attractors:
   ./cabean -compositional 2 -control <control> -sin <index of the source> -tin <index of the target> -rmID <file name> <model file>

4. Computation of the control for one pair of source and target attractors with constraints on attractors and perturbations:
   ./cabean -compositional 2 -control <control> -sin <index of the source> -tin <index of the target> -rmID <file name1> -rmPert <file name2> <model file>

5. Computation of the control for all combinations of source and target attractors:
   ./cabean -compositional 2 -control <control> -allpairs <model file>

6. Computation of the control for all combinations of source and target attractors with constraints on perturbations:
   ./cabean -compositional 2 -control <control> -allpairs -rmPert <file name> <model file>

7. Computation of the control for all combinations of source and target attractors with constraints on attractors:
   ./cabean -compositional 2 -control <control> -allpairs -rmID <file name> <model file>

   Note that for ASI control, the undesired attractors specified will be avoided as source, target and intermediate attractors; for OI, OT and OP control, the cases where undesired attractors are the source or the target will be skipped.

8. Computation of the control for all combinations of source and target attractors with constraints on attractors and perturbations:
   ./cabean -compositional 2 -control <control> -allpairs -rmID <file name1> -rmPert <file name2> <model file>

   Similar to the previous command line, for all the control methods, the cases where undesired attractors are the source or the target will be skipped; and for ASI control, the undesired attractors will also be avoided as intermediate attractors.

Target control

1. Computation of the target control for an attractor:
   ./cabean -compositional 2 -control <control> -tin <index of the target> <model file>
   '<control>' denotes one of the target control methods (ITC, TTC and PTC).

2. Computation of the target control for a phenotype:
   ./cabean -compositional 2 -control <control> -tmarker <file name> <model file>
   '<control>' denotes one of the target control methods (ITC, TTC and PTC).

4.3 Quickstart Guide

In this section, we use the example introduced in Section 3 to illustrate the basic usages of CABEAN. The model file is save as 'example.ispl' and it is placed in the directory of the executable file. Note that in the command line, <`model file`> represents the path of the model file and the directory of the executable file is used as the current directory. If the model file is not placed in the same directory with the executable file, replace "example.ispl" with the relative path of the model file in the command lines.

Attractor Detection

Before proceeding to the control part, attractors of the network are computed. CABEAN implements a decomposition-based attractor detection method [4] that can identify all the exact attractors of the network efficiently. The attractors of the toy example are computed with the following command line:
"./cabean -compositional 2 example.ispl"
The output of CABEAN is given in Figure 3. This network consists of six single-state attractors.

```
 Command line:  ./cabean -compositional 2 example.ispl
====== find attractor #1 :  1 states ======
:  7 nodes 1 leaves 1 minterms
0-0-1-1-0-0- 1
====== find attractor #2 :  1 states ======
:  7 nodes 1 leaves 1 minterms
0-0-1-1-1-1- 1
====== find attractor #3 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-0-0-0-0- 1
====== find attractor #4 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-0-0-1-0- 1
====== find attractor #5 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-1-1-0-0- 1
====== find attractor #6 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-1-1-1-1- 1
number of attractors = 6
time for attractor detection=0.003 seconds
```

Figure 3: Attractors of the example.

Minimal One-step Control

The command lines for the minimal OI, OT and OP control are similar. We choose the minimal OT control as a representative to show how to compute the control for

(1) one pair of source and target attractors and (2) all combinations of source and target attractors with and without practical constraints.

1. **Minimal OT control from A_6 to A_1.**
   Let us assume the source and target attractors of the toy example are A_6 and A_1, respectively. All the minimal OT control from A_6 to A_1 are computed using the following command line:
   "./cabean -compositional 2 -control OT -sin 6 -tin 1 example.ispl"
   The results are printed out in the terminal as shown in Figure 4. CABEAN first identifies all the exact attractors of the network, prints them in lexicographical order and then computes all the minimal OT control. We can see that there are in total eight minimal OT control sets that require two perturbations. Once the process is clear, henceforth, we will omit the attractor part in the output.

```
 Command line:  ./cabean -compositional 2 -control OT -sin 6 -tin 1
example.ispl
====== find attractor #1 :  1 states ======
:  7 nodes 1 leaves 1 minterms
0-0-1-1-0-0- 1
====== find attractor #2 :  1 states ======
:  7 nodes 1 leaves 1 minterms
0-0-1-1-1-1- 1
====== find attractor #3 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-0-0-0-0- 1
====== find attractor #4 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-0-0-1-0- 1
====== find attractor #5 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-1-1-0-0- 1
====== find attractor #6 :  1 states ======
:  7 nodes 1 leaves 1 minterms
1-1-1-1-1-1- 1
number of attractors = 6
time for attractor detection=0.002 seconds
=== ONE-STEP TEMPORARY SOURCE-TARGET CONTROL (DECOMP) ===
source - 6 target - 1
PATH 1 - #perturbations:  2
Control set:  x2=0 x6=0
PATH 2 - #perturbations:  2
Control set:  x2=0 x5=0
PATH 3 - #perturbations:  2
Control set:  x2=0 x4=0
PATH 4 - #perturbations:  2
Control set:  x2=0 x3=0
PATH 5 - #perturbations:  2
Control set:  x1=0 x6=0
PATH 6 - #perturbations:  2
Control set:  x1=0 x5=0
PATH 7 - #perturbations:  2
Control set:  x1=0 x4=0
PATH 8 - #perturbations:  2
Control set:  x1=0 x3=0
execution time for control = 0.005 seconds
```

Figure 4: Computation of the minimal OT control from A_6 to A_1.

2. **Minimal OT control from A_6 to A_1 with constraints on perturbations.** Suppose the node x_5 can not be perturbed from 1 to 0; the nodes x_4 and x_6 can not be perturbed in any way. The file for specifying the constraints, 'rmPert.txt', has been given in Section 3.2. It is also placed in the same directory with the executable binary file. We compute the minimal OT control under these constraints using the following command:
"./cabean -compositional 2 -control OT -sin 6 -tin 1 -rmPert rmPert.txt example.ispl"
The output of CABEAN is given in Figure 5. Compared to the results in

Figure 4, we can see that CABEAN only computes the control sets without undesired perturbations specified in the file 'rmPert.txt'.

```
 Command line:  ./cabean -compositional 2 -control OT -sin 6 -tin 1
-rmPert rmPert.txt example.ispl

... omit the attractors ...

=== ONE-STEP TEMPORARY SOURCE-TARGET CONTROL (DECOMP) ===
source - 6 target - 1
PATH 1 - #perturbations:  2
Control set:  x2=0 x3=0
PATH 2 - #perturbations:  2
Control set:  x1=0 x3=0
execution time for control = 0.002 seconds
```

Figure 5: Computation of the minimal OT from A_6 to A_1 with constraints on perturbations.

3. **Minimal OT control for all pairs of source and target attractors.**
   We use the following commands to compute the minimal OT control for all combinations of source and target attractors:
   "./cabean -compositional 2 -control OT -allpairs example.ispl"
   The output of CABEAN is shown in Figure 6. CABEAN first performs attractor detection and then compute the minimal OT control for each pair of source and target attractors. After traversing all combinations, the execution time and the number of perturbations for each case are summarised.

4. **Minimal OT control for all pairs of source and target attractors with constraints on attractors**
   Given some undesired attractors, the cases where the source and/or the target is undesired can be skipped by adding '-rmID <file name>' to the command line. Suppose the undesired attractors are A_2 and A_5, the specification file ('rmID.txt') has been given in Section 3.2 and it is save in the same directory with the executable binary file. The command line is:
   "./cabean -compositional 2 -control OT -allpairs -rmID rmID.txt example.ispl"
   All pairs of source and target attractors except for the cases, where the source or the target are A_2 or A_5, are computed.

In the command line, by replacing 'OT' with 'OI', 'OP', 'ASI', we can compute the minimal OI, OP, and ASI control for either one pair or all pairs of source and target attractors.

Attractor-based Sequential Control

Now we illustrate the computation of attractor-based sequential control from A_6 to A_1 with and without constraints on the attractors. The attractor-based control computes all the sequential control paths with at most k perturbations. The minimal number of perturbations required by the associated one-step control is used as the threshold k. We choose ASI control for the purpose of illustration, the threshold k is thus set as the number of perturbations required by the minimal OI control.

```
 Command line:  ./cabean -compositional 2 -control OT -allpairs
example.ispl
... omit the attractors ...

=== ONE-STEP TEMPORARY SOURCE-TARGET CONTROL (DECOMP) ===
--------------------
source - 1 target - 2
PATH 1 - #perturbations:  1
Control set:  x5=1
execution time for control = 0.001 seconds
----------------------
source - 1 target - 3
PATH 1 - #perturbations:  2
Control set:  x2=1 x4=0
PATH 2 - #perturbations:  2
Control set:  x2=1 x3=0
PATH 3 - #perturbations:  2
Control set:  x1=1 x4=0
PATH 4 - #perturbations:  2
Control set:  x1=1 x3=0
execution time for control = 0.002 seconds
... we omit some results ...
----------------------
source - 6 target - 5
PATH 1 - #perturbations:  1
Control set:  x5=0
execution time for control = 0.001 seconds
*************************************
print all the time costs:
0 0.001 0.002 0.002 0.001 0.001
0.001 0 0.003 0.002 0.001 0.001
0.001 0.002 0 0.001 0.001 0.001
0.005 0.004 0.001 0 0.001 0.001
0.002 0.001 0.001 0.001 0 0
0.003 0.001 0.001 0.001 0.001 0
print the number of driver nodes:
0 1 2 3 1 2
1 0 3 2 2 1
1 2 0 1 1 2
2 2 1 0 2 1
1 2 1 2 0 1
2 1 2 1 1 0
total time costs=0.046 seconds
```

Figure 6: Computation of the minimal OT control for all combinations of source and target attractors.

1. **ASI control from A_6 to A_1.** The computation of ASI control without any constraints are computed using the following command:
   "./cabean -compositional 2 -control ASI -sin 6 -tin 1 example.ispl"
   The output is given in Figure 7.

```
 Command line:  ./cabean -compositional 2 -control ASI -sin 6 -tin 1
example.ispl
... omit the attractors ...
=== ATTRACTOR-BASED SEQUENTIAL INSTANTANEOUS SOURCE-TARGET CONTROL
(DECOMP) ===
source - 6 target - 1
PATH 1 - #perturbations:  3
Sequence of the attractors:  6 -> 1
   STEP 1
     Control set 1:  x1=0 x2=0 x5=0
PATH 2 - #perturbations:  3
Sequence of the attractors:  6 -> 2 -> 1
   STEP 1
     Control set 1:  x1=0 x2=0
   STEP 2
     Control set 1:  x5=0
PATH 3 - #perturbations:  3
Sequence of the attractors:  6 -> 5 -> 1
   STEP 1
     Control set 1:  x5=0
   STEP 2
     Control set 1:  x1=0 x2=0
execution time of control=0.004 seconds
```

Figure 7: Computation of the ASI control from A_6 to A_1.

2. **ASI control from A_6 to A_1 with constraints on perturbations.** Suppose the node x_5 can not be perturbed from 1 to 0; the nodes x_4 and x_6 can not be perturbed either from 1 to 0 or the opposite. We can compute ASI control without undesired perturbations using the following command line:
   "./cabean -compositional 2 -control ASI -sin 6 -tin 1 -rmPert rmPert.txt example.ispl"
   The results are given in Figure 8. We can see that there does not exist any one-step instantaneous control paths satisfying the constraints and thus the threshold is set to 0.

```
 Command line:  ./cabean -compositional 2 -control ASI -sin 6 -tin 1
-rmPert rmPert.txt example.ispl
... omit the attractors ...
=== ATTRACTOR-BASED SEQUENTIAL INSTANTANEOUS SOURCE-TARGET CONTROL
(DECOMP) ===
source - 6 target - 1
There does not exist any direct path satisfying the constraints.
The threshold on the number of perturbations is 0.
```

Figure 8: Computation of the ASI control from A_6 to A_1 with constraints on perturbations.

3. **ASI control from A_6 to A_1 with constraints on attractors.** Suppose attractors A_2 and A_5 are undesired attractors as defined in Section 3.2 and the file is saved as 'rmID.txt'. We compute the ASI control bypassing undesired attractors using the following command:

   "./cabean -compositional 2 -control ASI -sin 6 -tin 1 -rmID rmID.txt example.ispl"

   The output is given in Figure 9.

```
 Command line:  ./cabean -compositional 2 -control ASI -sin 6 -tin 1
-rmID rmID.txt example.ispl
... omit the attractors ...
=== ONE-STEP TEMPORARY SOURCE-TARGET CONTROL (DECOMP) ===
IDs of undesired attractors
2 5
---------------------
source - 6 target - 1
PATH 1 - #perturbations:  3
Sequence of the attractors:  6 -> 1
   STEP 1
     Control set 1:  x1=0 x2=0 x5=0
execution time of control=0.002 seconds
```

Figure 9: Computation of ASI from A_6 to A_1 with constraints on attractors.

Target Control

Suppose attractor A_1 is the target attractor. We compute TTC to A_1 using the following command:

"./cabean -compositional 2 -control TTC -tin 1 example.ispl"

The output is given in Figure 10.

```
 Command line:  ./cabean -compositional 2 -control TTC -tin 1
example.ispl
... omit the attractors ...
=== TEMPORARY TARGET COTNROL (DECOMP) ===
*****************************************
TARGET ATTRACTOR #1
*****************************************
Control set 1:  x1=0 x3=0
Control set 2:  x2=0 x3=0
Control set 3:  x1=0 x4=0
Control set 4:  x2=0 x4=0
Control set 5:  x1=0 x5=0
Control set 6:  x2=0 x5=0
Control set 7:  x1=0 x6=0
Control set 8:  x2=0 x6=0
Time for temporary target control = 0.01 seconds
```

Figure 10: Computation of TTC to A_1.

Suppose we are interested in the phenotype P, where x_6 is 1. The specification file, saved as 'targetPhenotype.txt', is given below.

```
node, value
x6, 1
```

Figure 11: Specification of a phenotype

We compute TTC to P using the following command:
"./cabean -compositional 2 -control TTC -tmarker targetPhenotype.txt example.ispl"
The output is given in Figure 12.

```
 Command line:  ./cabean -compositional 2 -control TTC -tmarker
targetPhenotype.txt example.ispl
... omit the attractors ...
=== TEMPORARY TARGET COTNROL (DECOMP) ===
*****************************************
TARGET ATTRACTOR #2 #6
*****************************************
Control set 1:  x1=0 x5=1
Control set 2:  x2=0 x5=1
Control set 3:  x3=1 x5=1
Control set 4:  x4=1 x5=1
Time for temporary target control = 0.007 seconds
```

Figure 12: Computation of TTC to P.

5 Case study

Hematopoiesis is the process through which mature blood cells are manufactured. The myeloid differentiation network is constructed to model the differentiation process of common myeloid progenitors (CMPs) into four types of mature blood cells [2]. , viz. megakaryocytes, erythrocytes, granulocytes and monocytes [2]. This network consists of 11 transcription factors as shown in Figure 13. In this section, we show how to use CABEAN to analyse the myeloid differentiation network step by step.

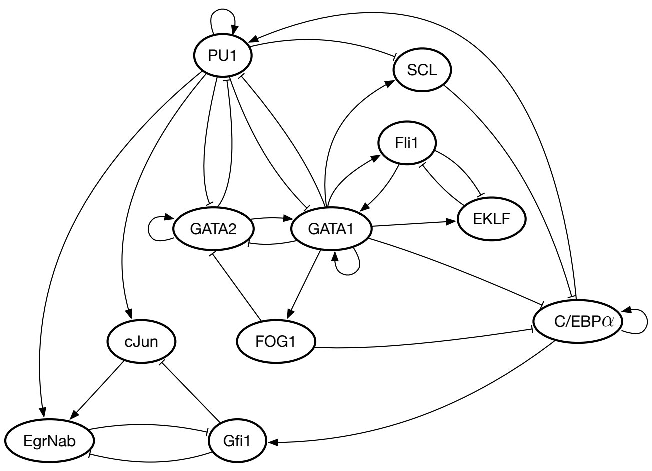


Figure 13: Structure of the myeloid network.

5.1 Model File

The Boolean functions can be found in the original work [2]. The model files of the myeloid differentiation network under BoolNet and ISPL format are shown in Figure 14 and Figure 15, respectively.

```
targets, factors
GATA2, GATA2 & (!(GATA1 & FOG1)) & (!PU1)
GATA1, (GATA1 | GATA2 | Fli1) & (!PU1)
FOG1, GATA1
EKLF, GATA1 & (!Fli1)
Fli1, GATA1 & (!EKLF)
SCL, GATA1 & (!PU1)
CEBPA, CEBPA & (!(GATA1 & FOG1 & SCL))
PU1, (CEBPA| PU1) & (!(GATA1 | GATA2))
cJun, PU1 & (!Gfi1)
EgrNab, (PU1 & cJun) & (!Gfi1)
Gfi1, CEBPA & (!EgrNab)
```

Figure 14: The model file of the myeloid differentiation network under BoolNet format.

5.2 Attractor Detection

Since ISPL format is preferable, we use 'myeloid.ispl' for the illustration. To analyse the myeloid differentiation network, we first compute its attractors with the following command line:
"./cabean -compositional 2 myeloid.ispl"
The output of CABEAN is given in Figure 16. With our attractor detection method [4],

```
Agent M
   Vars:
      GATA2:  boolean;
      GATA1:  boolean;
      FOG1:  boolean;
      EKLF: boolean;
      Fli1:  boolean;
      SCL: boolean;
      CEBPA: boolean;
      PU1:  boolean;
      cJun:  boolean;
      EgrNab:  boolean;
      Gfi1:  boolean;
   end Vars
   Actions = none;
   Protocol:
      Other:  none;
   end Protocol
   Evolution:
      GATA2=true if GATA2&( (GATA1&FOG1))&( PU1)=true;
      GATA2=false if GATA2&( (GATA1&FOG1))&( PU1)=false;
      GATA1=true if (GATA1|GATA2|Fli1)&( PU1)=true;
      GATA1=false if (GATA1|GATA2|Fli1)&( PU1)=false;
      FOG1=true if GATA1=true;
      FOG1=false if GATA1=false;
      EKLF=true if GATA1&( Fli1)=true;
      EKLF=false if GATA1&( Fli1)=false;
      Fli1=true if GATA1&( EKLF)=true;
      Fli1=false if GATA1&( EKLF)=false;
      SCL=true if GATA1&( PU1)=true;
      SCL=false if GATA1&( PU1)=false;
      CEBPA=true if CEBPA&( (GATA1&FOG1&SCL))=true;
      CEBPA=false if CEBPA&( (GATA1&FOG1&SCL))=false;
      PU1=true if (CEBPA|PU1)&( (GATA1|GATA2))=true;
      PU1=false if (CEBPA|PU1)&( (GATA1|GATA2))=false;
      cJun=true if PU1&( Gfi1)=true;
      cJun=false if PU1&( Gfi1)=false;
      EgrNab=true if (PU1&cJun)&( Gfi1)=true;
      EgrNab=false if (PU1&cJun)&( Gfi1)=false;
      Gfi1=true if CEBPA&( EgrNab)=true;
      Gfi1=false if CEBPA&( EgrNab)=false;
   end Evolution
end Agent

InitStates
      M.GATA2=true or M.GATA2=false;
end InitStates
```

Figure 15: The model file of the myeloid differentiation network under ISPL format.

we identify six single-state attractors given in Table 2. It has been validated that expressions of four attractors correspond to microarray expression profiles of megakaryocytes (A_5), erythrocytes (A_6), granulocytes (A_3) and monocytes (A_4) [2]. The attractor A_2 with the activation of PU1, cJun and EgrNab might be caused by pathological alterations [2] and the all-zero (A_1) attractor is an attractor, where all the

```
 Command line:  ./cabean -compositional 2 myeloid.ispl
======= find attractor #1 :  1 states ========
:  12 nodes 1 leaves 1 minterms
0-0-0-0-0-0-0-0-0-0-0- 1
======= find attractor #2 :  1 states ========
:  12 nodes 1 leaves 1 minterms
0-0-0-0-0-0-0-1-1-1-0- 1
======= find attractor #3 :  1 states ========
:  12 nodes 1 leaves 1 minterms
0-0-0-0-0-0-1-1-0-0-1- 1
======= find attractor #4 :  1 states ========
:  12 nodes 1 leaves 1 minterms
0-0-0-0-0-0-1-1-1-1-0- 1
======= find attractor #5 :  1 states ========
:  12 nodes 1 leaves 1 minterms
0-1-1-0-1-1-0-0-0-0-0- 1
======= find attractor #6 :  1 states ========
:  12 nodes 1 leaves 1 minterms
0-1-1-1-0-1-0-0-0-0-0- 1
number of attractors = 6
time for attractor detection=0.004 seconds
```

Figure 16: Computation of attractors of the myeloid-differentiation network.

	GATA2	GATA1	FOG1	EKLF	Fli1	SCL	C/EBPA	PU1	cJun	EgrNab	Gfi1
A_1	0	0	0	0	0	0	0	0	0	0	0
A_2	0	0	0	0	0	0	0	1	1	1	0
A_3	0	0	0	0	0	0	1	1	0	0	1
A_4	0	0	0	0	0	0	1	1	1	1	0
A_5	0	1	1	0	1	1	0	0	0	0	0
A_6	0	1	1	1	0	1	0	0	0	0	0

Table 2: Attractors of the myeloid differentiation network.

nodes have a value of '0'.

We take the conversion from megakaryocytes (A_5) to monocytes (A_4) as an example to show the performance of CABEAN. Note that the all-zero (A_1) attractor does not have a biological interpretation and mature erythrocytes (A_6) in mammals do not have cell nucleus, therefore we do not consider these two attractors as intermediate attractors for attractor-based sequential control.

5.3 Minimal One-step Instantaneous Control

We compute the minimal OI control from A_5 to A_4 with the following command line:
"./cabean -compositional 2 -control OI -sin 5 -tin 4 myeloid.ispl"
The output of CABEAN is given in Figure 17.

```
 Command line:  ./cabean -compositional 2 -control OI -sin 5 -tin 4
myeloid.ispl
... omit the attractors ...

====== ONE-STEP INSTANTANEOUS SOURCE-TARGET CONTROL (DECOMP) ======
source - 5 target - 4
PATH 1 - #perturbations:  5
Control set:  GATA1=0 EgrNab=1 CEBPA=1 PU1=1 cJun=1
execution time of control = 0.001 seconds
```

Figure 17: Minimal OI control of the myeloid differentiation network.

5.4 Minimal One-step Temporary Control

We compute the minimal OT control from A_5 to A_4 using the following command line:
"./cabean -compositional 2 -control OT -sin 5 -tin 4 myeloid.ispl"
The output of CABEAN is given in Figure 18.

```
 Command line:  ./cabean -compositional 2 -control OT -sin 5 -tin 4
myeloid.ispl
... omit the attractors ...

====== ONE-STEP TEMPORARY SOURCE-TARGET CONTROL (DECOMP) ======
source - 5 target - 4
Number of paths:  2
PATH 1 - #perturbations:  3
Control set:  EgrNab=1 CEBPA=1 PU1=1
PATH 2 - #perturbations:  3
Control set:  GATA1=0 EgrNab=1 CEBPA=1
execution time for control = 0.011 seconds
```

Figure 18: Minimal OT control of the myeloid differentiation network.

5.5 Minimal One-step Permanent Control

We compute the minimal OP control from A_5 to A_4 using the following command line:
"./cabean -compositional 2 -control OP -sin 5 -tin 4 myeloid.ispl"
The output of CABEAN is given in Figure 19.

```
 Command line:  ./cabean -compositional 2 -control OP -sin 5 -tin 4
myeloid.ispl
... omit the attractors ...

====== ONE-STEP PERMANENT SOURCE-TARGET CONTROL (DECOMP) ======
source - 5 target - 4
PATH 1 - #perturbations:  3
Control set:  EgrNab=1 CEBPA=1 PU1=1
PATH 2 - #perturbations:  3
Control set:  GATA1=0 EgrNab=1 CEBPA=1
execution time for control = 0.01 seconds
```

Figure 19: Minimal OP control of the myeloid differentiation network.

5.6 Attractor-based Sequential Instantaneous Control

As mentioned before, attractors A_1 and A_6 cannot play the role of intermediate attractors. The specification file ('rmID.txt') that defines the undesired intermediate attractors is shown below.

```
1,6
```

Under this condition, we compute the minimal ASI control from A_5 to A_4 with the command:
"./cabean -compositional 2 -control ASI -rmID rmID.txt -sin 5 -tin 4 myeloid.ispl"
The output of CABEAN is given in Figure 20.

```
 Command line:  ./cabean -compositional 2 -control ASI -rmID rmID.txt
-sin 5 -tin 4 myeloid.ispl
... omit the attractors ...
=== ATTRACTOR-BASED SEQUENTIAL INSTANTANEOUS SOURCE-TARGET CONTROL
(DECOMP) ===
IDs of undesired attractors
1 6
source - 5 target - 4
PATH 1 - #perturbations:  5
Sequence of the attractors:  5 -> 4
   STEP 1
     Control set 1:  GATA1=0 EgrNab=1 CEBPA=1 PU1=1 cJun=1
PATH 2 - #perturbations:  3
Sequence of the attractors:  5 -> 2 -> 4
   STEP 1
     Control set 1:  GATA1=0 PU1=1
   STEP 2
     Control set 1:  CEBPA=1
execution time of control=0.006 seconds
```

Figure 20: ASI control of the myeloid differentiation network.

5.7 Attractor-based Sequential Temporary Control

We compute the AST control without passing by A_1 and A_6 using the following command line:
"./cabean -compositional 2 -control AST -rmID rmID.txt -sin 5 -tin 4 myloid.ispl"
The output of CABEAN is given in Figure 21.

```
 Command line:  ./cabean -compositional 2 -control AST -rmID rmID.txt
-sin 5 -tin 4 myeloid.ispl
... omit the attractors ...
=== ATTRACTOR-BASED SEQUENTIAL TEMPORARY SOURCE-TARGET CONTROL
(DECOMP) ===
IDs of undesired attractors
1 6
source - 5 target - 4
PATH 1 - #perturbations:  3
Sequence of the attractors:  5 -> 4
   STEP 1
     Control set 1:  EgrNab=1 CEBPA=1 PU1=1
     Control set 2:  GATA1=0 EgrNab=1 CEBPA=1
PATH 2 - #perturbations:  2
Sequence of the attractors:  5 -> 2 -> 4
   STEP 1
     Control set 1:  PU1=1
   STEP 2
     Control set 1:  CEBPA=1
execution time of control=0.029 seconds
```

Figure 21: AST control of the myeloid differentiation network.

5.8 Attractor-based Sequential Permanent Control

We compute the ASP control without passing by A_1 and A_6 using the following command line:
"./cabean -compositional 2 -control ASP -rmID rmID.txt -sin 5 -tin 4 myloid.ispl"
The output of CABEAN is given in Figure 22.

```
 Command line:  ./cabean -compositional 2 -control ASP -rmID rmID.txt
-sin 5 -tin 4 myeloid.ispl
... omit the attractors ...
=== ATTRACTOR-BASED SEQUENTIAL PERMANENT SOURCE-TARGET CONTROL
(DECOMP) ===
IDs of undesired attractors
1 6
source - 5 target - 4
PATH 1 - #perturbations:  3
Sequence of the attractors:  5 -> 4
   STEP 1
     Control set 1:  EgrNab=1 CEBPA=1 PU1=1
     Control set 2:  GATA1=0 EgrNab=1 CEBPA=1
PATH 2 - #perturbations:  2
Sequence of the attractors:  5 -> 2 -> 4
   STEP 1
     Control set 1:  PU1=1
   STEP 2
     Control set 1:  CEBPA=1
execution time of control=0.035 seconds
total time costs=0.035 seconds
```

Figure 22: ASP control of the myeloid differentiation network.

5.9 Temporary Target Control

We compute TTC to A_4 using the following command line:
"./cabean -compositional 2 -control TTC -tin 4 myeloid.ispl"
The output of CABEAN is given in Figure 23.

```
 Command line:  ./cabean -compositional 2 -control TTC -tin 4
myeloid.ispl
... omit the attractors ...

======== TEMPORARY TARGET COTNROL (DECOMP) ========
**************************************************
TARGET ATTRACTOR #4
**************************************************
Control set 1:  EgrNab=1 CEBPA=1 PU1=1
Control set 2:  CEBPA=1 PU1=1 Gfi1=0
Time for temporary target control = 0.012 seconds
```

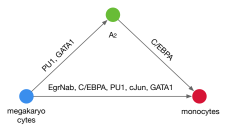
Figure 23: TTC of the myeloid differentiation network.

5.10 Evaluation of the Control Results

The control paths (OI, OT, OP, ASI, AST and ASP) from megakaryocytes (A_5) to monocytes (A_4) of the myeloid differentiation network are summarised in Figure **??**. The arrows from megakaryocytes to monocytes represent the one-step control paths

(OI, OT or OP). The sequential paths with different types of perturbations use attractor A_2 as the intermediate. OT and OP have the same results. AST and ASP also identify the same control paths.

We can see that the minimal OI control requires the activation of EgrNab, C/EBPA, PU1, cJun and the inhibition of GATA1 (Fig. **??**); while OT or OP can achieve the goal by either (1) the activation of EgrNab, C/EBPA and PU1; or (2) the activation of EgrNab and C/EBPA, together with the inhibition of GATA1 (Fig. **??**). All the sequential paths consist of two steps, where A_2 is adopted as an intermediate attractor. For the first step, ASI activates PU1 and inhibits GATA1, while AST or ASP only needs to activate PU1. When the network converges to the fifth attractor, all the three methods require to activate C/EBPA. After that, the network will evolve spontaneously to the target attractor monocytes. Fig. **??** shows that AST and ASP are able to identify a path with only two perturbations, while ASI requires at least three perturbations.



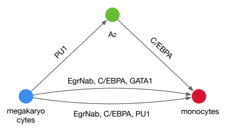


Figure 24: Control of the myeloid differentiation network.

The efficacy of the identified sequential temporary/permanent path is confirmed by the predictions in [2]. According to the expression profiles, both PU1 and C/EBPA are not expressed in MegE lineage (megakaryocytes and erythrocytes), while they are expressed in GM lineage (monocytes and granulocytes). In this network, no regulator can activate C/EBPA and PU1 is primarily activated by C/EBPA. Therefore, C/EBPA has to be altered externally to reprogram MegE lineage to GM lineage. However, more perturbations are necessary to accurately reach the monocytes lineage. Sustained activation of PU1 and the absence of C/EBPA guide the network to A_2, the expression of which differs with monocytes only in C/EBPA [2].

Bibliography

[1] Martin Hopfensitz, Christoph Müssel, and Hans A Kestler. A short introduction to the boolnet package.

[2] Jan Krumsiek, Carsten Marr, Timm Schroeder, and Fabian J Theis. Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network. *PLOS ONE*, 6(8):e22649, 2011.

[3] Hugues Mandon, Cui Su, Stefan Haar, Jun Pang, and Loïc Paulevé. Sequential reprogramming of Boolean networks made practical. In *Proc. 17th International Conference on Computational Methods in Systems Biology*, volume 11773 of *LNCS*, pages 3–19. Springer, 2019.

[4] A. Mizera, J. Pang, H. Qu, and Q. Yuan. Taming asynchrony for attractor detection in large Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(1):31–42, 2019.

[5] Aurélien Naldi. BioLQM: a java toolkit for the manipulation and conversion of logical qualitative models of biological networks. *Frontiers in Physiology*, 9:1605, 2018.

[6] S. Paul, C. Su, J. Pang, and A. Mizera. A decomposition-based approach towards the control of Boolean networks. In *Proc. 9th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 11–20. ACM Press, 2018.

[7] Soumya Paul, Cui Su, Jun Pang, and Andrzej Mizera. An efficient approach towards the source-target control of Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019. accepted.

[8] Cui Su and Jun Pang. A dynamics-based approach for the target control of Boolean networks. In *Proc. 11th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM Press, 2020. accepted.

[9] Cui Su and Jun Pang. Sequential temporary and permanent control of Boolean networks. In *Proc. 18th International Conference on Computational Methods in Systems Biology*, LNCS. Springer, 2020. accepted.

[10] Cui Su, Soumya Paul, and Jun Pang. Controlling large Boolean networks with temporary and permanent perturbations. In *Proc. 23rd International Symposium on Formal Methods*, volume 11800 of *LNCS*, pages 707–724. Springer-Verlag, 2019.