

Aeon Manual

1	What does Aeon do	0
2	Getting Aeon running	0
2.1	Running pre-compiled binaries	1
2.2	Building from source	1
2.3	Startup.....	1
3	Model description	1
4	Graphical user interface	2
4.1	Aeon Main Page.....	2
4.2	Computation module.....	2
4.3	Compute Engine module.....	3
4.4	Import/Export module.....	3
4.5	Model Editor module.....	4
4.6	Controllable Editor module.....	4
4.7	Phenotype Editor module.....	5
4.8	Visual Options module.....	5
4.9	Results module.....	6
4.10	Inner Tab system	7
5	Demonstration of computation process.....	8
5.1	Control Computation.....	8
7	Computation Model format and update function syntax	10

November 2024 <https://sybila.fi.muni.cz/>

1 What does Aeon do

As a member of BioDivine suite, Aeon (Analysis & Exploration of Networks) is a parallel tool for creating, editing, and analysing parametrised Boolean network models; specifically, it provides means of analysis of model's bifurcations — qualitative changes in behaviour, which are originating in, typically small, changes of parameters. Details on the underlying theory can be found in [1].

2 Getting Aeon running

The tool implementation consists of two components: the *compute engine*, and the web based, user-facing GUI application (the *client*). A typical use of the tool requires a local installation of the compute engine, which is accessed from the client. The client can be also stored locally, or hosted remotely, with no change in functionality between the two cases. The online version of the client is accessible from <https://biodivine.fi.muni.cz/aeon>; for offline use, the client application can be downloaded from <https://github.com/sybila/biodivine-aeon-client>. The client application can be used to create and edit parametric models without the compute engine being installed. The client does not connect to the internet. The engine can be obtained as a pre-compiled executable (for all major desktop platforms) or as a Rust source code. Because the client is accessing the engine via http connection in which the engine acts as a server, it is possible to access the engine remotely, assuming sufficient network configuration—this is useful when the computation is delegated to a suitable powerful hardware.

Client

online access	biodivine.fi.muni.cz/aeon/
offline download	github.com/sybila/biodivine-aeon-client/
Engine	
source, executables	github.com/sybila/biodivine-aeon-server/releases/

2.1 Running pre-compiled binaries

Pre-compiled executables for multiple platforms are available at <https://github.com/sybila/biodivine-aeon-server/releases>. After downloading and running the corresponding file, the engine will be accessible from the client application and ready for use. The relevant executables can be also downloaded through the links listed in the client application under the *compute engine* panel, described in Section 4.2. Preparing the executable on Linux:

```
$ unzip aeon-compute-engine-linux.zip && chmod +x aeon-compute-engine
```

2.2 Building from source

The engine source code, written in the Rust programming language and licensed under the MIT License, is freely available for download. To compile the software, one needs to install the Rust toolchain – *rustup*, and download the actual source code.

- *rustup* – <https://www.rust-lang.org/tools/install>
- *Compute engine* – <https://github.com/sybila/biodivine-aeon-server>

When the Rust toolchain is installed following the instructions on its website, the engine can be compiled using the

```
$ cargo +nightly build
```

command in the root of the directory. After successful compilation, running

```
$ cargo run
```

will start up the engine.

2.3 Startup

By default, the engine uses the localhost address and the port 8000 to run on. If the port is available, the engine will report the address and the port number on which it is running.

```
Rocket has launched from http://localhost:8000
```

The default server address and port will work in most cases; however, should the automatic assignment fail, manual configuration is possible through the environment variables `AEON_ADDR` and `AEON_PORT`. For example, setting a different port number would look like this (on Linux/Mac):

```
$ export AEON_PORT=3485
```

After the engine has been properly configured and it's up and running, the client will automatically establish a connection on its startup. If it is already running in the web browser, clicking on the *Connect* button under the *compute engine* panel will link the two, and the tool will be ready to be used.

3 Model description

The Aeon does use parametrized Boolean network models. A Boolean network can be seen as a directed graph.

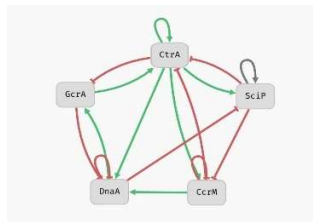
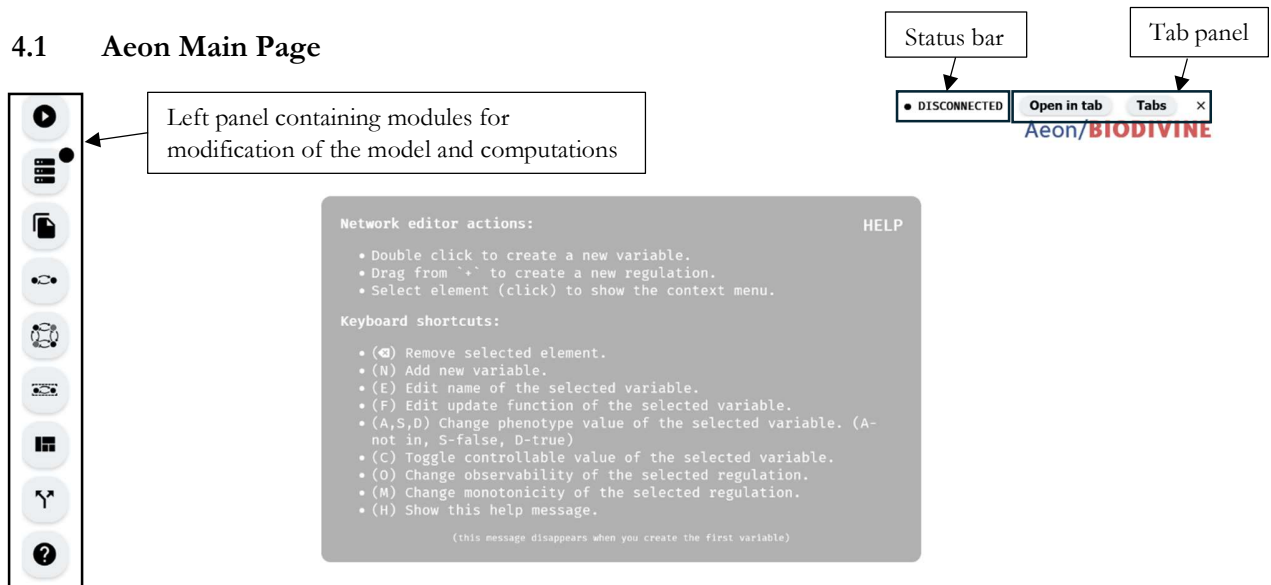


Figure 1: A simple Boolean network as displayed in Aeon– model adopted from [3].

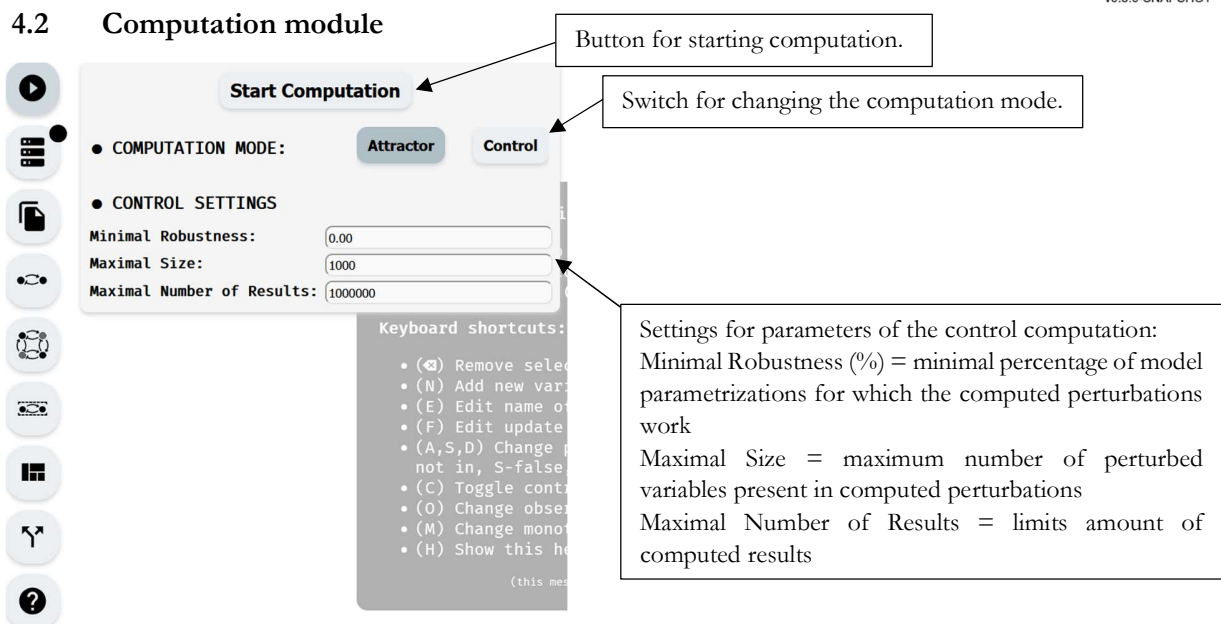
4 Graphical user interface

The client, running in a web browser, provides a user-friendly graphical interface, that enables one to create, edit, and visualise Boolean network models on the one hand, and allows for interfacing with the engine, supervising the computation, and visualisation of the results on the other. Models are drawn and displayed on the large editor canvas. At any time, pressing and holding the H key will display the help window.

4.1 Aeon Main Page



4.2 Computation module



4.3 Compute Engine module

The screenshot shows the 'Compute Engine' window. It features a status bar at the top with a green dot and the word 'CONNECTED'. Below this, it says 'Computation: (none)'. There is a 'Disconnect' button with a cloud icon. A message says 'No engine running? Download binary:' followed by buttons for 'Windows', 'MacOS', and 'Linux'. A keyboard shortcuts list is visible on the right side of the window.

Annotations:

- Address of the running Compute Engine (set by default at `http://localhost:8000`)
- Connect/Disconnect button. To be able to connect Compute Engine needs to run on the same machine.
- Compute Engine status.
- Computation status Present only when the Compute Engine is connected.
- Download links for Compute Engines by platform.

4.4 Import/Export module

The screenshot shows the 'Model File' window. It has two main sections: 'Import' and 'Export'. The 'Import' section includes buttons for 'Last model', 'Browser local storage', '.AEON', 'Simple text format', '.SBML', 'Standard SBML L3', '.BNET', and 'Boolnet text format'. The 'Export' section includes buttons for '.AEON', 'Simple text format', '.SBML (parametrized)', 'Parametrized model', '.SBML (instantiated)', 'Witness model', '.BNET', and 'Boolnet text format'. There is also an 'Example Models' section with buttons for 'G2A', 'Cell Division', 'Orlando', 'Budding Yeast', 'G2B', 'Cell Division', 'Irons', and 'Budding Yeast'.

Annotations:

- Model import buttons for different formats.
- Model export buttons for different formats.
- Import buttons for example models.

4.5 Model Editor module

Model name

Show/hide model description.

Information about the model

Add new variable into the model.

Add new variable into the model.

Information about regulators of the variable

Update function of the variable (syntax of the update function defined in the section 5.)

Delete variable from the model

Find variable inside of the graphical representation of the model.

Toggle phenotype status of the variable.
 Grey = not present in the phenotype
 Green = present in the phenotype as true
 Red = present in the phenotype as false

Variable name

Show/hide variable information.

Toggle variable controllability.
 Yellow = variable is controllable
 Grey = variable is not controllable

OVERVIEW

Variables: 5 Parameter space size: 2^{48}
 Regulations: 15 State space size: 2^5
 Max. in-degree: 4 Max. out-degree: 5
 Explicit parameters: (none)

VARIABLES

REGULATORS

Regulator	Regulation	Observable	Activation
CtrlA	->	observable	activation
GcrA	->	observable	activation
CcrM	-	observable	inhibition
SciP	-	observable	inhibition

UPDATE FUNCTION

$f_{CtrlA}(\dots)$

Possible instantiations: 114

4.6 Controllable Editor module

Controllable/Uncontrollable variables counters

Select/unselect buttons.
 Left – toggles selected (unselects selected, selects unselected)
 Middle – selects all variables
 Right – unselects all variables.
 Functionality of these buttons is restricted to filtered data (for example if Filter bar contains CtrlA, then middle button only selects CtrlA)

Filter bar
 Name of every variable in the filter has to be separated by ` ` (example – CtrlA, GcrA)
 Filter also works for filtering names starting with string (for example if we write `C` into the filter, then table shows all variables starting with C)

Variable controllability indicator
 Yellow = variable is controllable
 Grey = variable is not controllable

Variable name Filter bar add button
 Adds name of the variable into the Filter bar

Variable row
 Variable can be selected/unselected by clicking/dragging over variable row.

Controllable/Uncontrollable switches
 Yellow = makes all selected variables controllable
 Grey = makes all selected variables not controllable

OVERVIEW

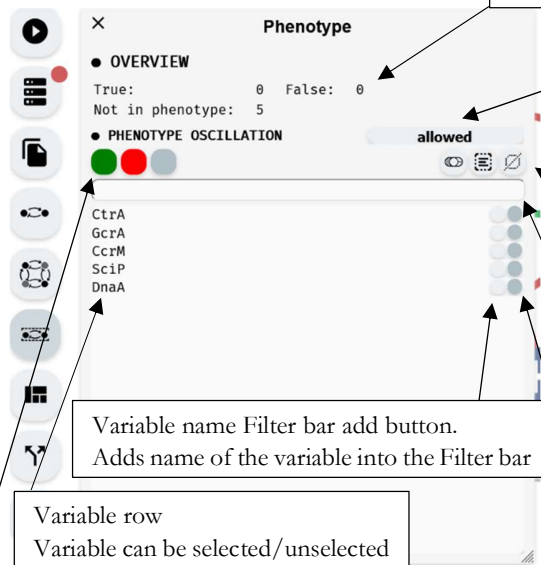
Controllable: 5 Uncontrollable: 0

Variables

Variable	Controllable
CtrlA	Yellow
GcrA	Yellow
CcrM	Yellow
SciP	Yellow
DnaA	Yellow

4.7 Phenotype Editor module

Phenotype status variable counters



Phenotype oscillation toggle
 allowed = set phenotype may oscillate
 required = set phenotype must oscillate
 forbidden = set phenotype cannot oscillate
 Phenotype oscillates when it repeatedly appears after some time but doesn't stay fixed to the required value forever.

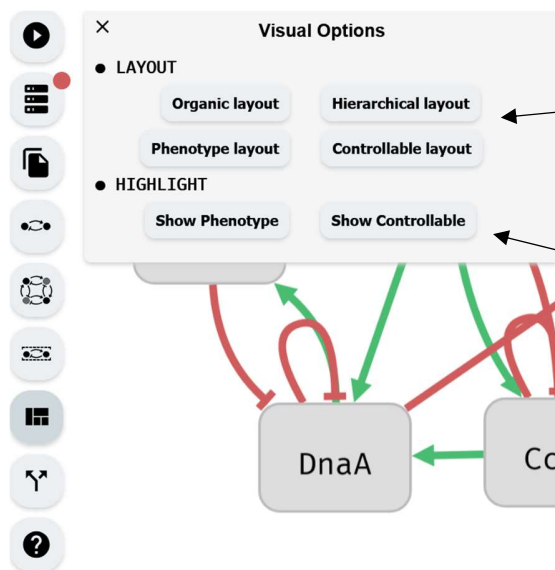
Select/unselect buttons.
 Left – toggles selected (unselects selected, selects unselected)
 Middle – selects all variables.
 Right – unselects all variables.
 Functionality of these buttons is restricted to filtered data (for example if Filter bar contains CtrA, then middle button only selects CtrA)

Filter bar
 Name of every variable in the filter has to be separated by ` ` (example – CtrA, GcrA)
 Filter also works for filtering names starting with string (for example if we write `C` into the filter, then table shows all variables starting with C)

Variable phenotype status indicator
 Grey = not present in the phenotype
 Green = present in the phenotype as true
 Red = present in the phenotype as false

Phenotype status switches
 Grey = removes all selected variables from the phenotype
 Green = adds all selected variables into the phenotype as true
 Red = adds all selected variables into the phenotype as false

4.8 Visual Options module



Buttons for applying different layouts of nodes to the graphical interpretation of the model.

Highlights nodes in the graphical representation of the model according to some property
 Show Phenotype – Highlights borders of nodes by its phenotype value (Grey = not present in the phenotype, Green = present in the phenotype as true, Red = present in the phenotype as false).
 Show Controllable – Highlights nodes which are controllable with yellow color.

4.9 Results module

Information about computed perturbations.

Elapsed = how long did the computation take

Number of Param = number of parametrizations of the model

Number of Pert = number of calculated perturbations

Minimal Size = minimal number of variables found in perturbation

Maximal Robustness = maximal robustness (for what % of parametrizations perturbation works) found in perturbation

Oscillation = oscillation status of the phenotype

Table visualization button

Opens new inner tab with calculated perturbations visualized in the form of table.

If there is a high number of perturbations may cause performance issues (when opened gives option to export table into .csv file)

Control results

Witness count

Number of parametrizations which's behavior belongs to this attractor class.

Witness Button

Opens new browser tab with one random parametrization of the model which's behavior belongs to this attractor class.

Attractor explorer button

Opens new inner tab with Attractor explorer of the current attractor behavior class.

Attractor explorer button

Opens new inner tab with Attractor explorer of the current attractor class.

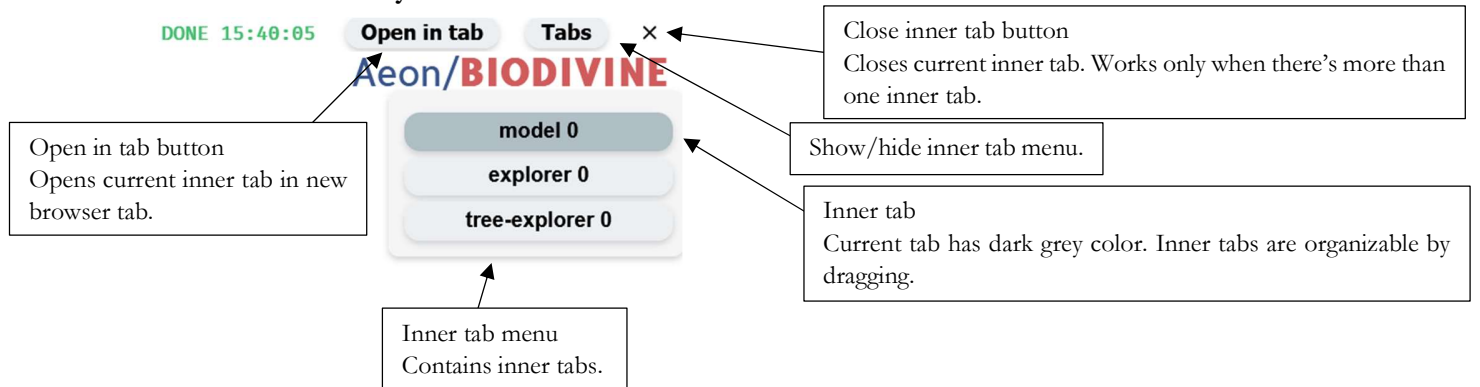
Attractor behavior class

Representation of how the attractor behaves.

- ⇌ Disorder = attractor shows complex behavior
- Oscillation = attractor cycles between states
- Stability = attractor stabilizes into one state

4.10 Inner Tab system

DONE 15:40:05

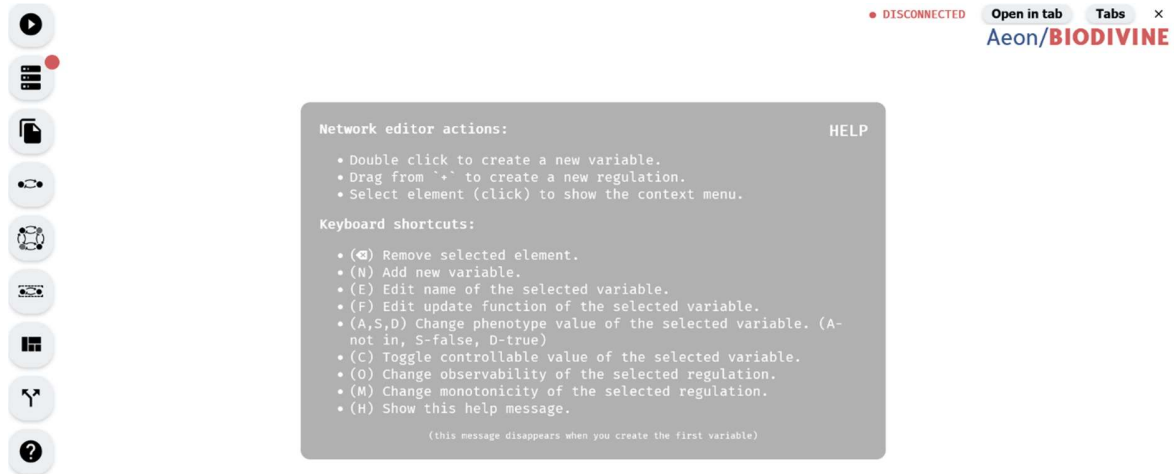


5 Demonstration of computation process

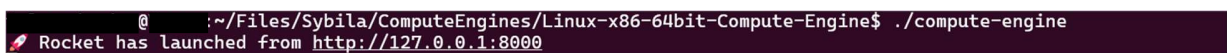
5.1 Attractor Analysis

5.2 Control Computation

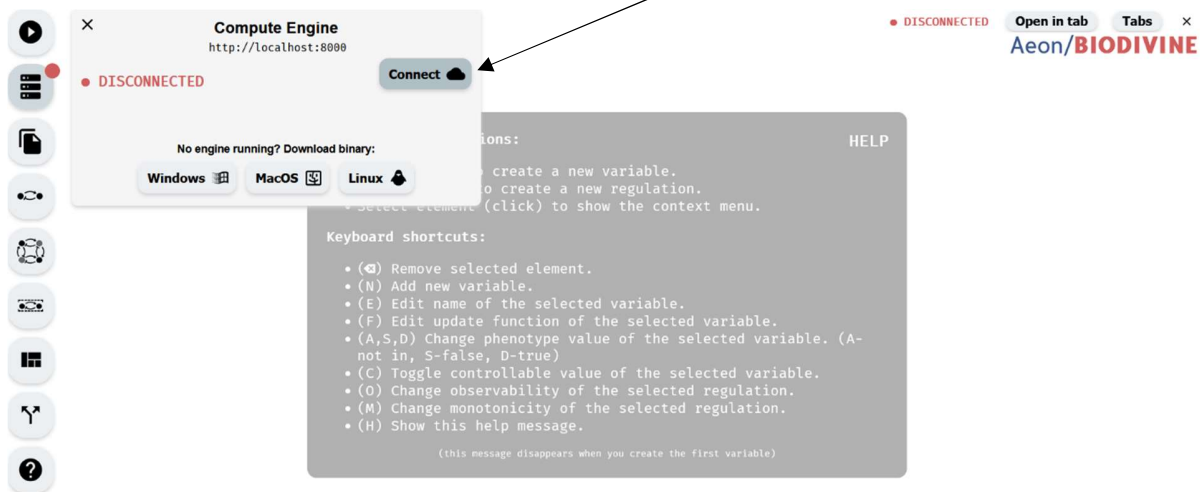
- 1) Start Biodivine/Aeon Online Tool client as described in the chapter 2 of the manual.



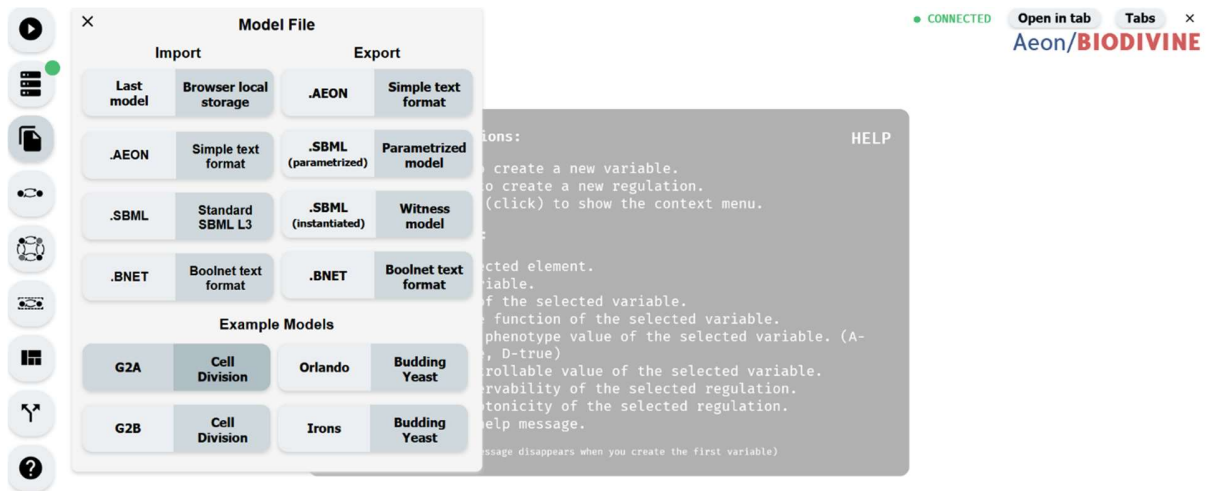
- 2) Start Biodivine/Aeon Online Tool Compute Engine as described in the chapter 2 of the manual.



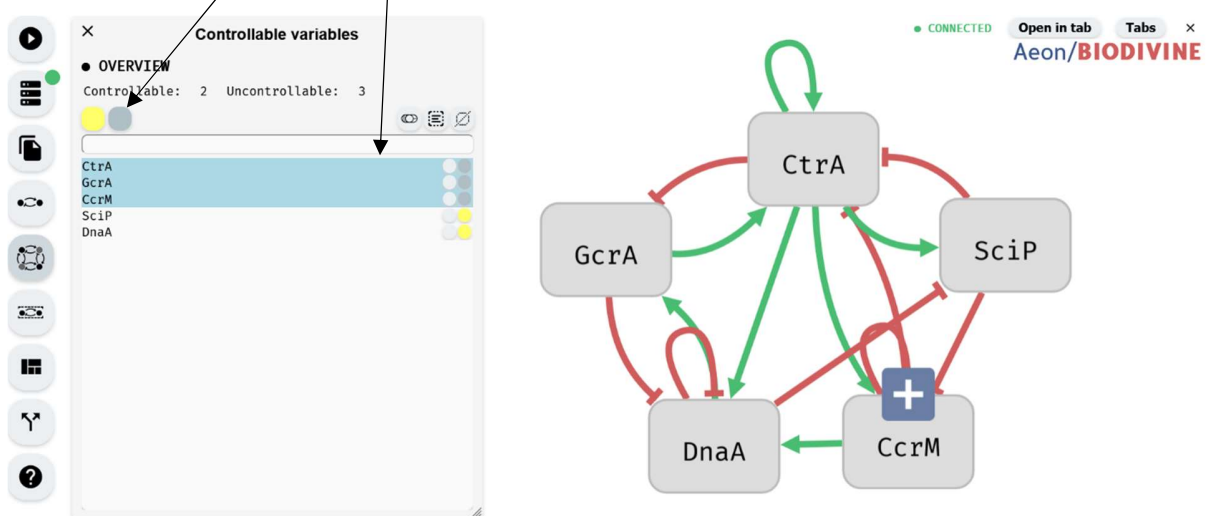
- 3) Connect compute engine to the client by clicking the connect/disconnect button in the Compute Engine Module



- 4) Import model with the Import/Export module. (we will import G2A example model)



- 5) Set controllable variables with the Controllable Editor module. (In this case we want variables CtrA, GcrA, CcrM as uncontrollable and SciP and DnaA as controllable)
Initially, all variables were set to "controllable." We selected the variables we wanted to make "uncontrollable" by clicking on their corresponding rows. Then, we changed their controllability status to "uncontrollable" by clicking the grey button.



- 6) Set phenotype with the Phenotype Editor module

7 Computation Model format and update function syntax

Models are in this format:

```
Aeon file ::= Regulation | Update fn decl | Meta | Control Stat | Aeon file \n Aeon file
Update fn decl ::= $ Name : Update fn
Meta ::= # Key : Value
Regulation ::= Name  $\sqsubset$  Arrow  $\sqsubset$  Name
Arrow ::= Kind | Kind?
Kind ::= -> | -| | -?

Control Stat ::= #!control: Name : VarControll, VarPhen
VarControll ::= true | false
VarPhen ::= true | false | null

Update fn ::= true | false | Name | Uninterpreted fn | !Update fn | (Update fn Op Update fn)
Op ::= & | | | => | <=>
Uninterpreted fn ::= Name(Parameters)
Parameters ::= Name | Parameters, Parameters
```

Only names of the can be used as function parameters.

References

- [1]
- [2] Nikola Beneš et al. “Formal Analysis of Qualitative Long-Term Behaviour in Parametrised Boolean Networks”. In: *Formal Methods and Software Engineering (ICFEM 2019)*. Springer, 2019, pp. 353–369.
- [3] Claudine Chaouiya et al. “SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools”. In: *BMC systems biology* 7.1 (2013), p. 135.
- [4] Ismael Sánchez-Osorio, Carlos A. Hernández-Martínez, and Agustino Martínez-Antonio. “Modeling Asymmetric Cell Division in *Caulo bacter crescentus* Using a Boolean Logic Approach”. In: *Asymmetric Cell Division in Development, Differentiation and Cancer*. Ed. by Jean-Pierre Tassan and Jacek Z. Kubiak. Cham: Springer International Publishing, 2017, pp. 1–21.