Aeon Manual

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.

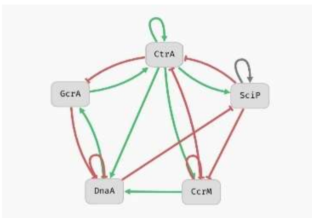
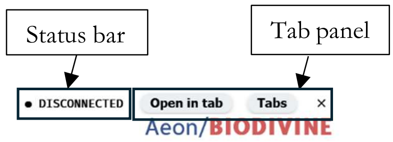


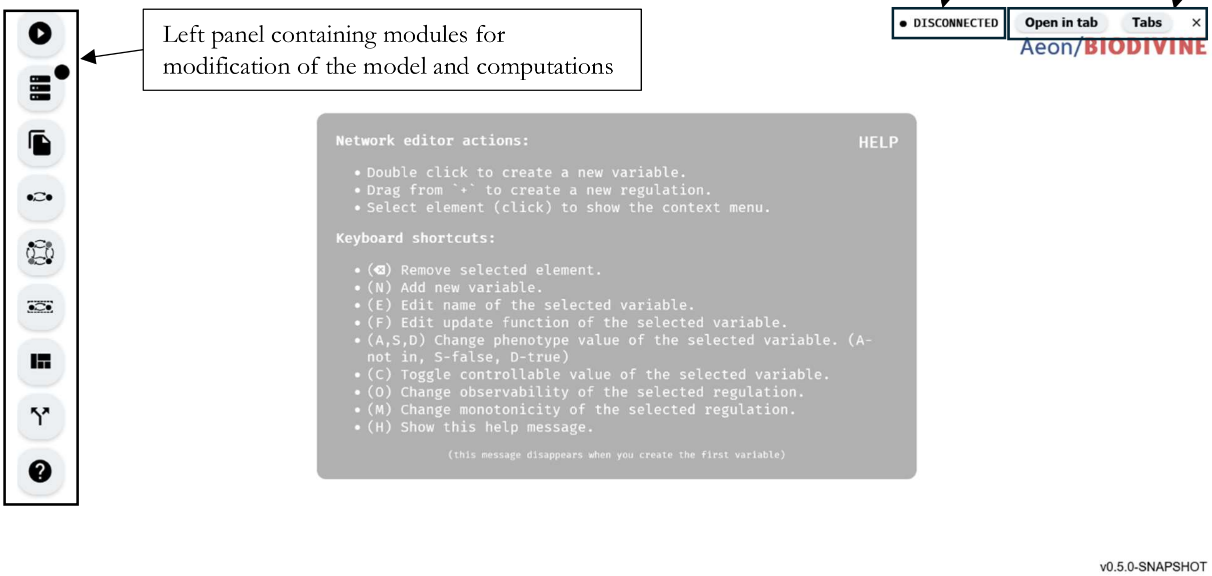
Figure 1: A simple Boolean network as displayed in Aeon– model adopted from [3].

4 Graphical user interface

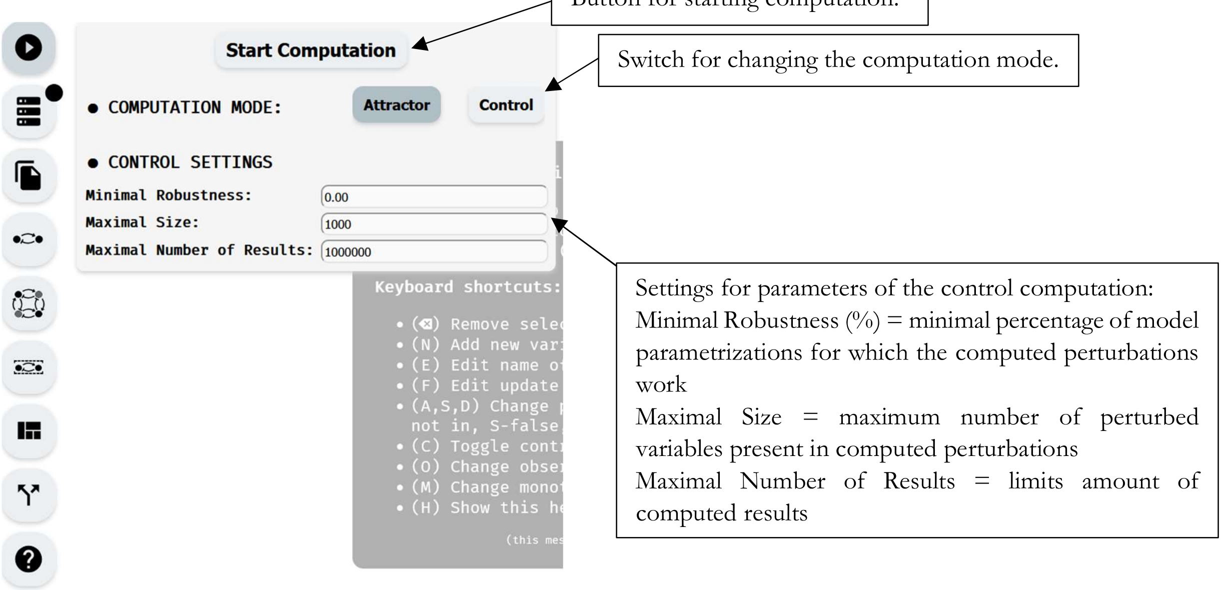
The client, running in a web browser, provides a user-friendly graphical interface, that enables one to create, edit, and visualise Boolean network models on the one hand, and allows for interfacing with the engine, supervising the computation, and visualisation of the results on the other. Models are 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

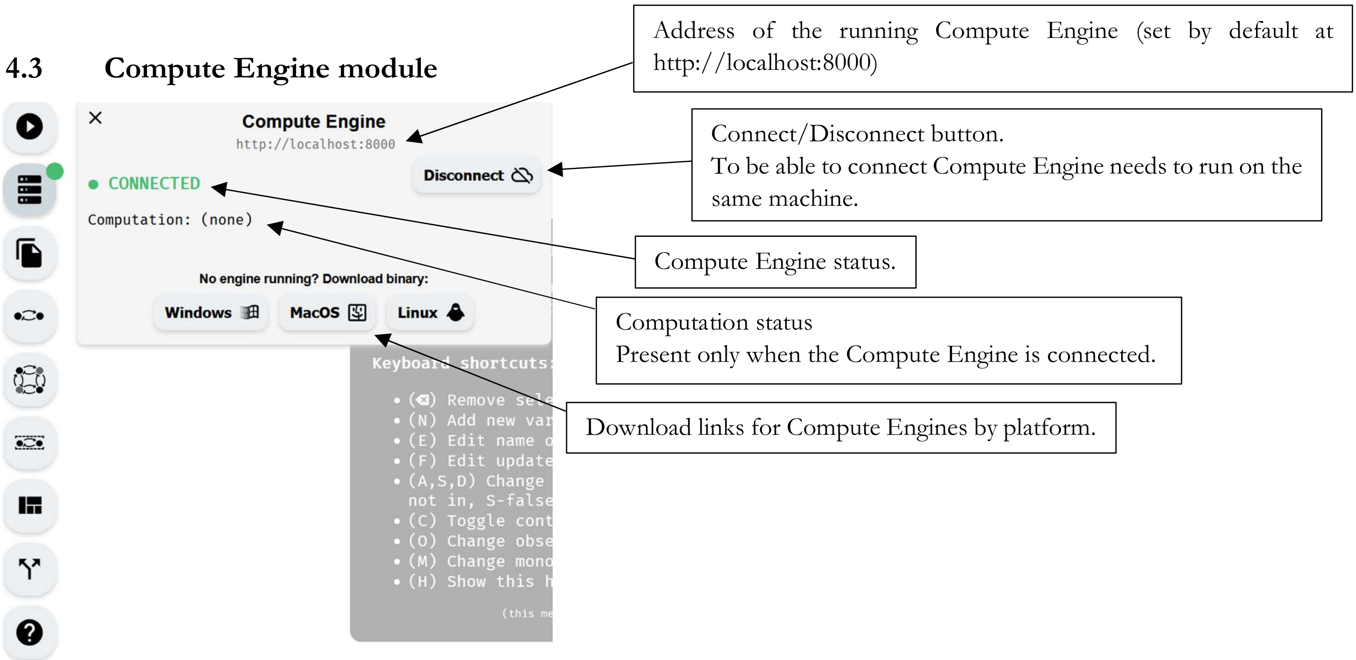




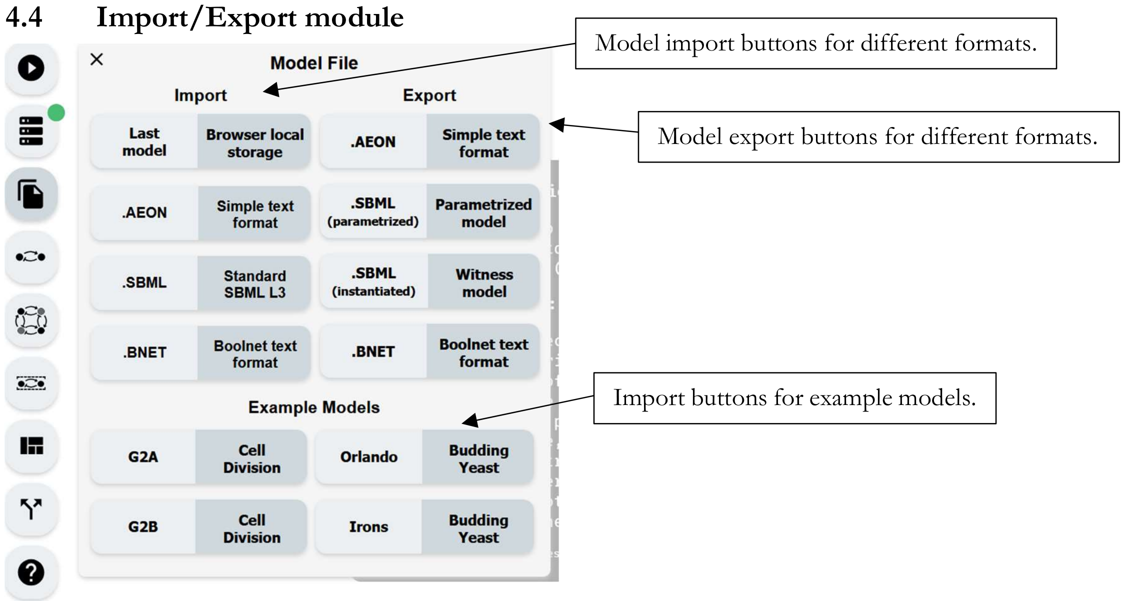
4.2 Computation module



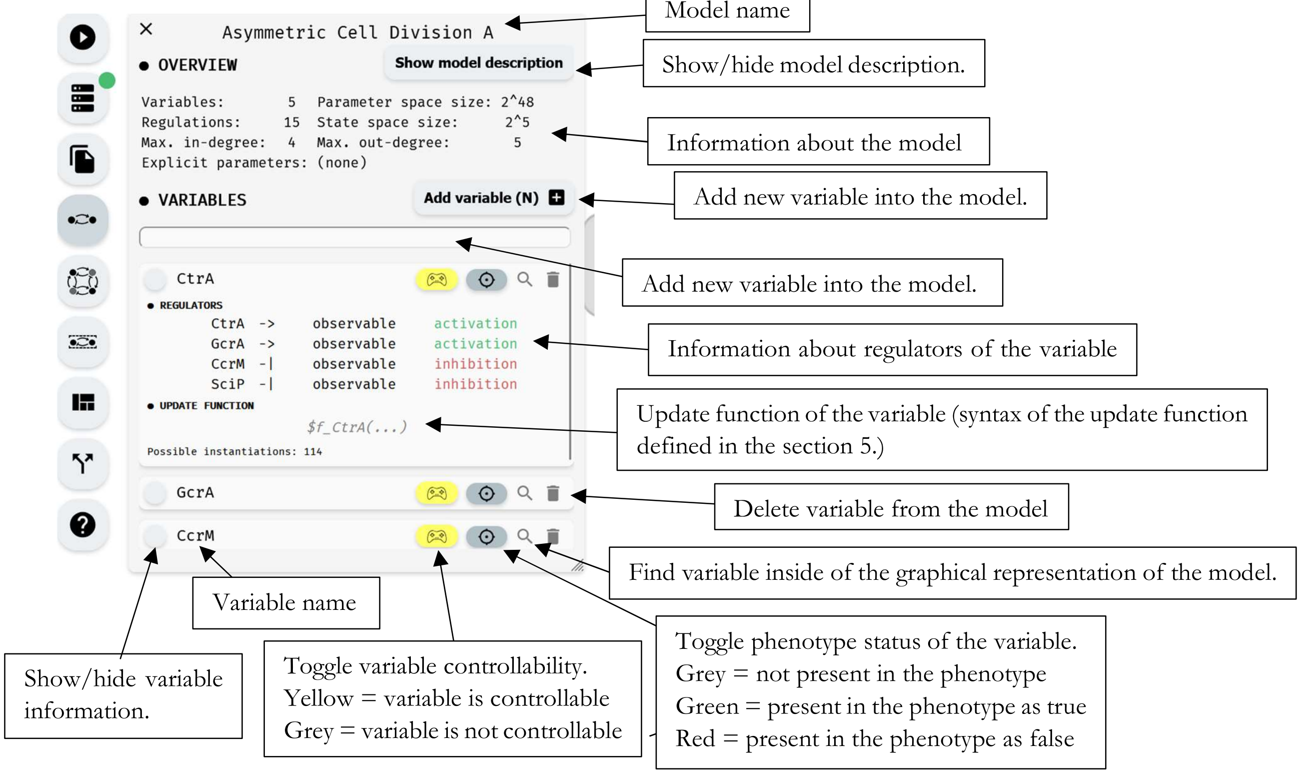
4.3 Compute Engine module



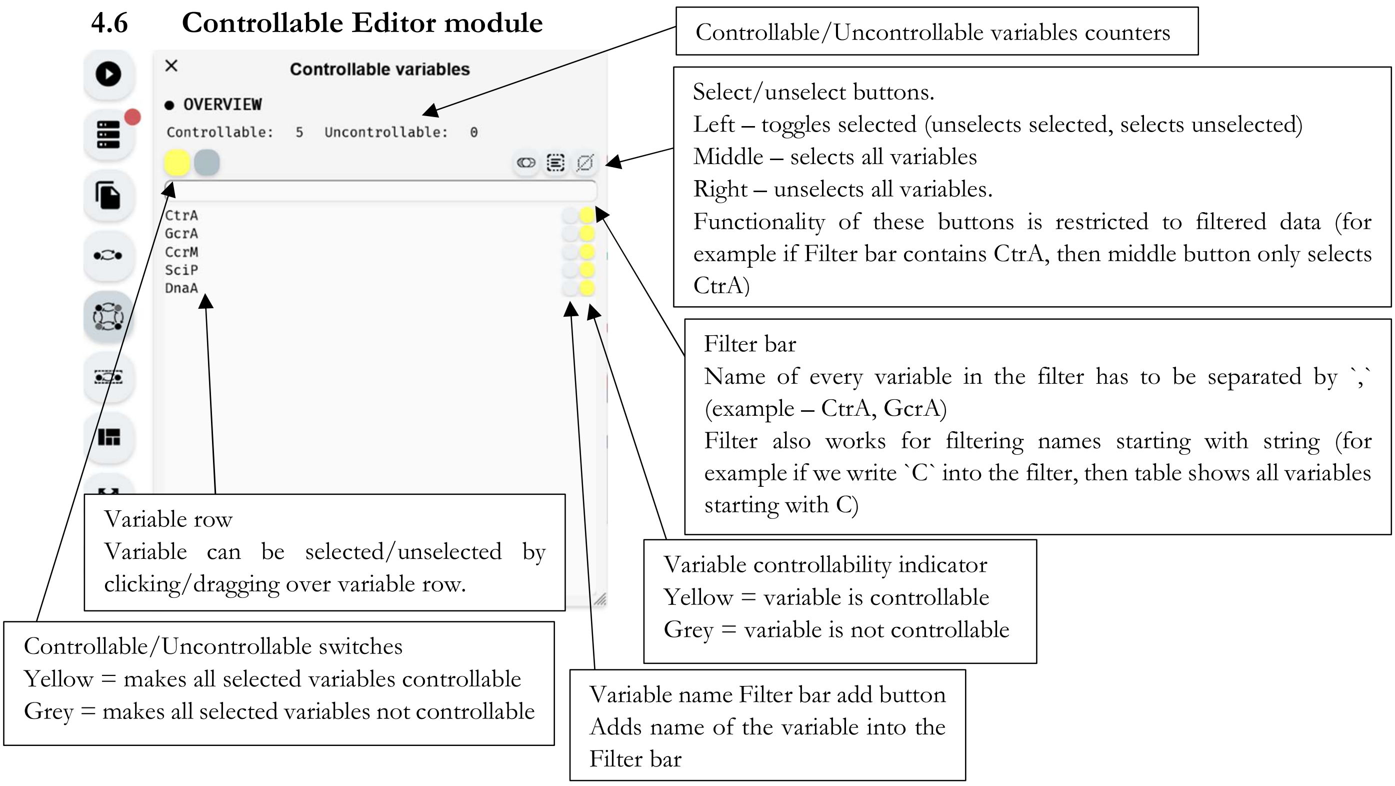
4.4 Import/Export module



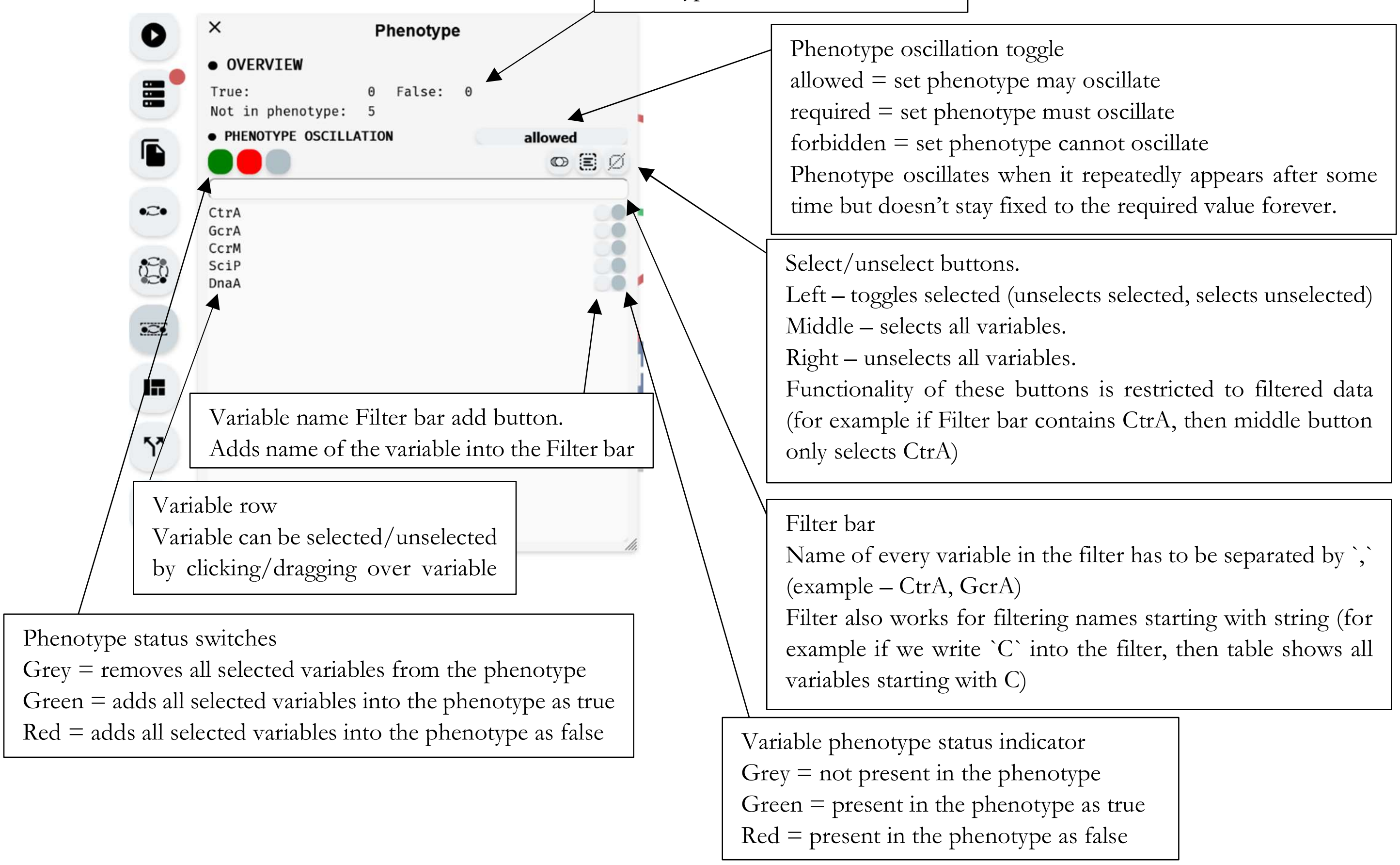
4.5 Model Editor module



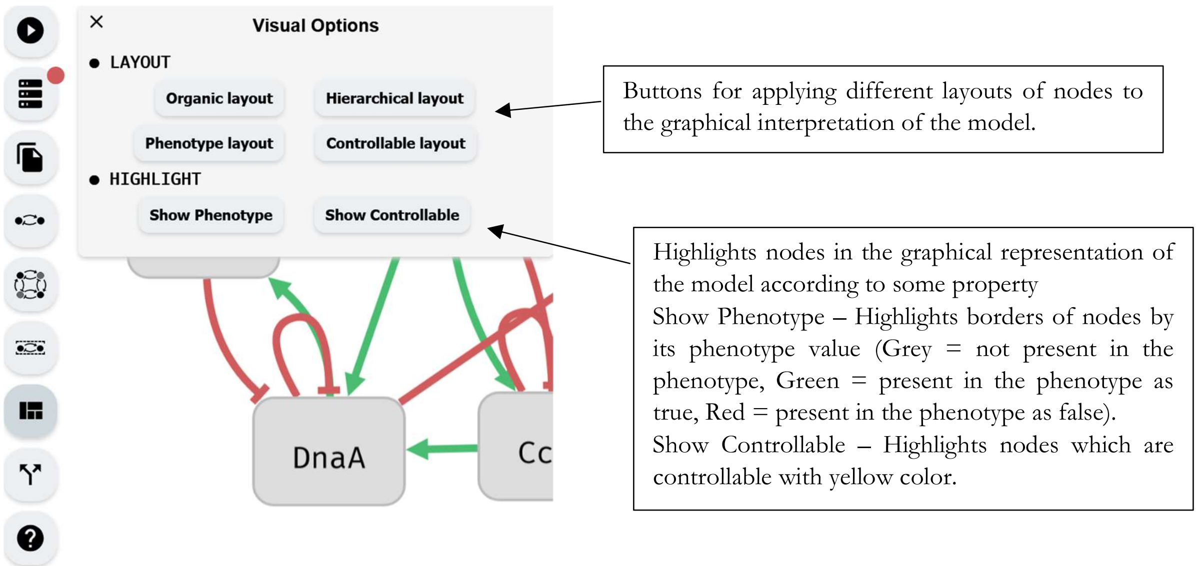
4.6 Controllable Editor module



4.7 Phenotype Editor module

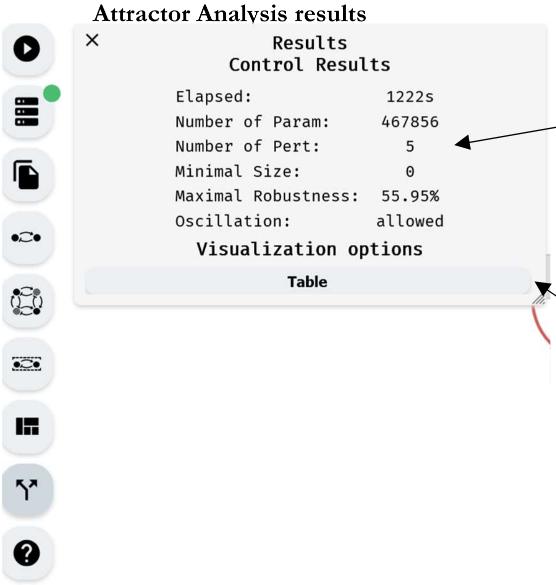


4.8 Visual Options module



4.9 Results module

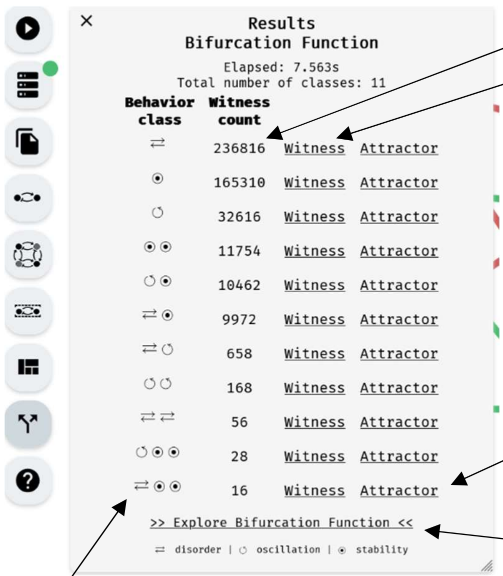
Attractor Analysis results



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.

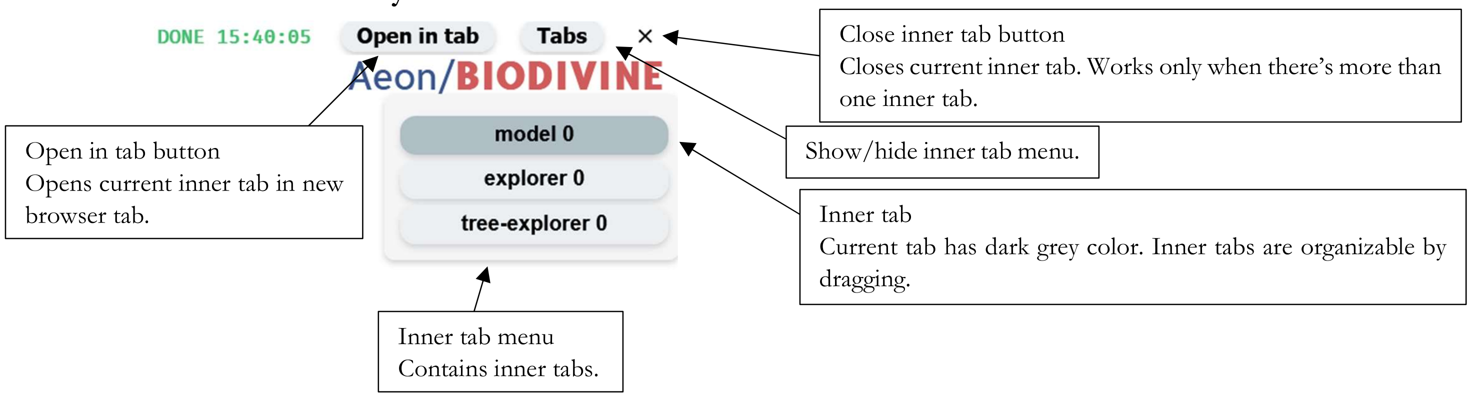
Witness Button
Opens new browser tab with one random parametrization of the model which's behavior belongs to this attractor class.

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

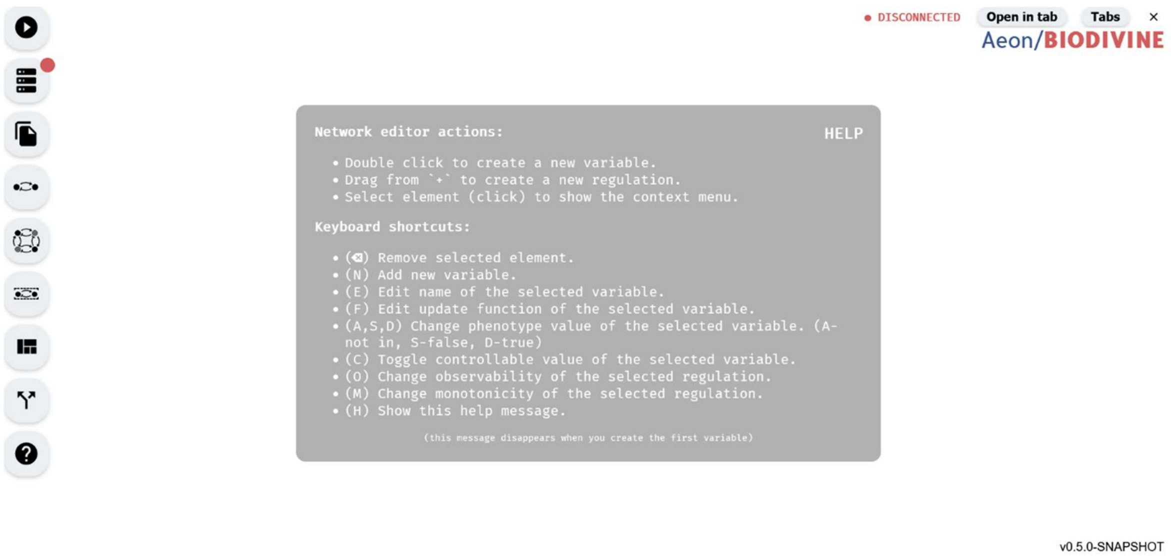


5 Demonstration of computation process

5.1 Attractor Analysis

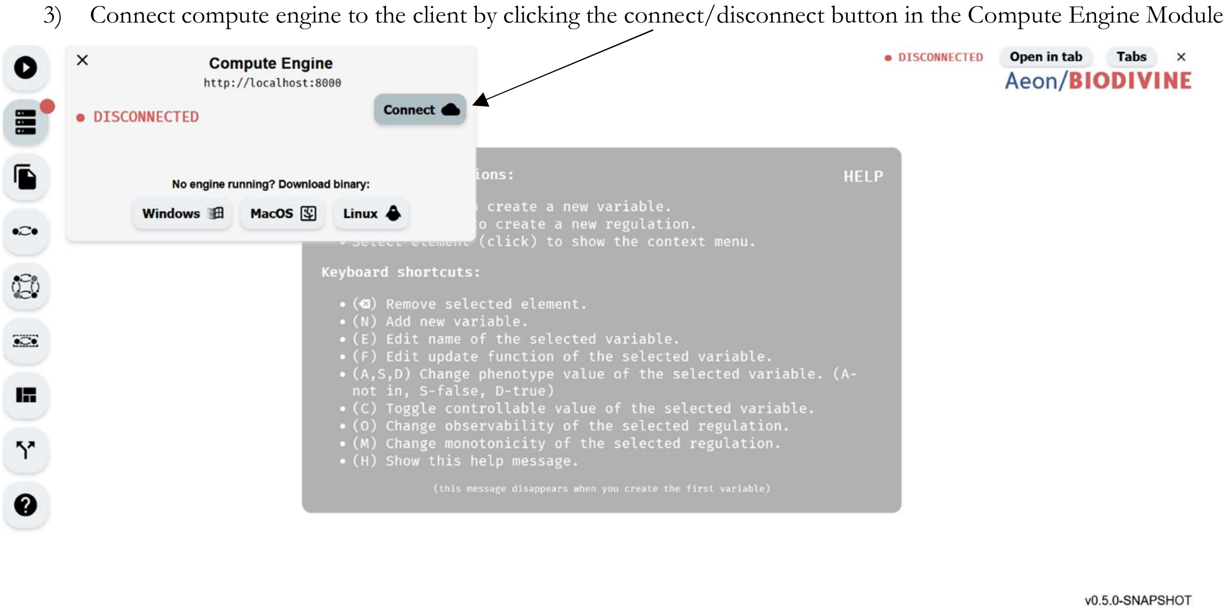
5.2 Control Computation

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

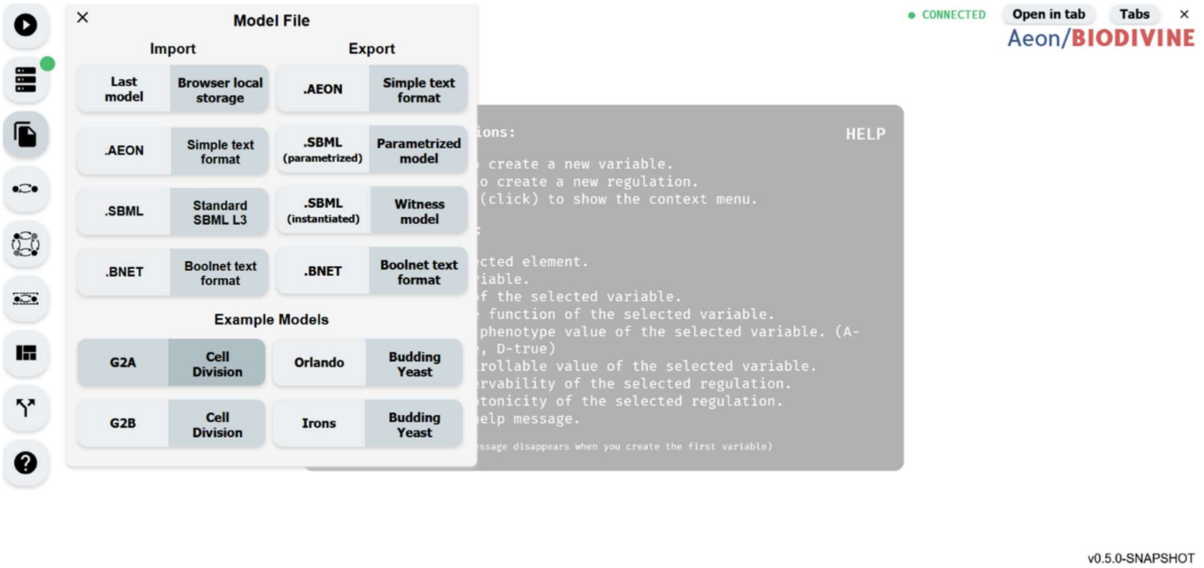


2) Start Biodivine/Aeon Online Tool Compute Engine as described in the chapter 2 of the manual.

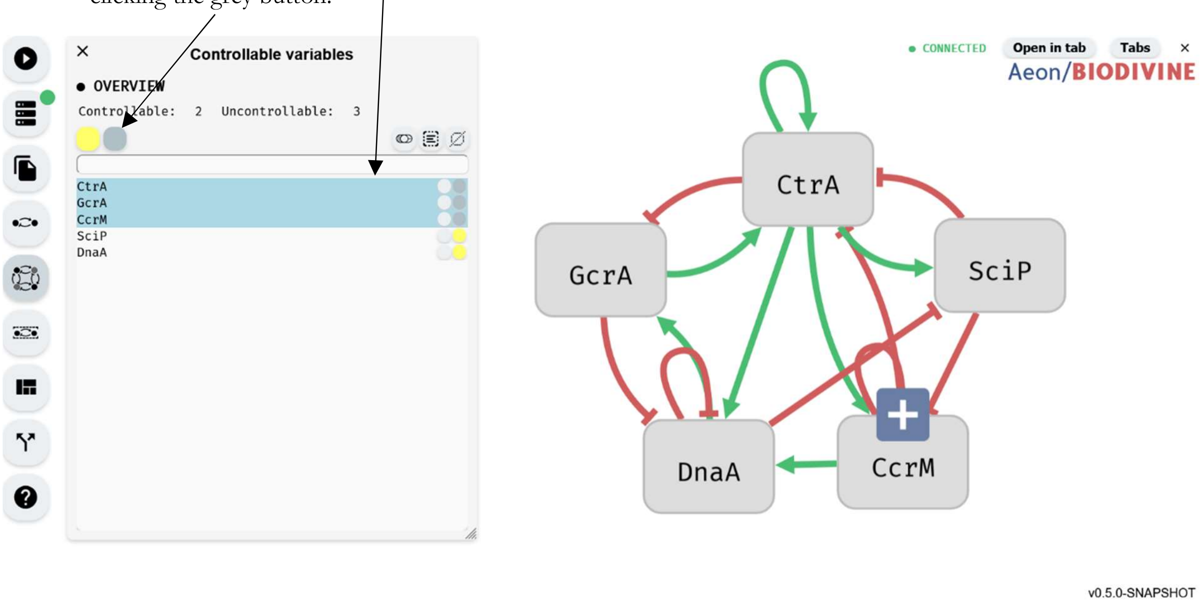
3) Connect compute engine to the client by clicking the connect/disconnect button in the Compute Engine Module



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*␣*Arrow*␣*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*e | *Parameters*, *Parameters*

Only names of the can be used as function parameters.

References

[1]

[2] Nikola Benešet 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 bactercrescentus 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.