Aeon Manual

January 2020 `https://sybila.fi.muni.cz/`

1 What does Aeon do

AsamemberofBioDivinesuite,Aeon(Analysis& ExplorationofNetworks)isaparallel tool for creating, editing, and analysing parametrised Boolean network models; specifically,itprovidesmeansofanalysisofmodel'sbifurcations—qualitativechangesinbehaviour, 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 webbased,user-facingGUIapplication(the*client*). Atypicaluseofthetoolrequiresalocalinstallation 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-compiledexecutablesformultipleplatformsareavailableat`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 `ld` command in the root of the directory. After successful compilation, running will start up the engine. [redacted]

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
```

Thedefaultserveraddressandportwillworkinmostcases; however,shouldtheautomaticassignmentfail,manualconfigurationispossiblethroughtheenvironmentvariables`AEON_ADDR` and`AEON_PORT`.Forexample, settingadifferentportnumberwouldlooklikethis(onLinux/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, clickingon the *Connect* buttonunder the *compute engine* panelwill link the two, and thetool will be ready to be used.

3 Model description

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

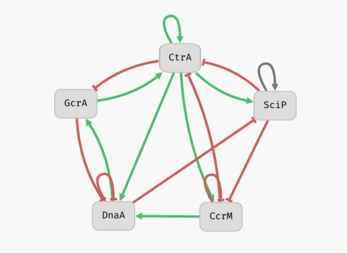
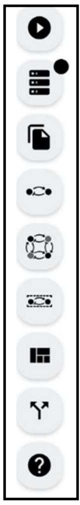


Figure 1: A simple Boolean network as displayed in Aeon– model adopted from [3].

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.

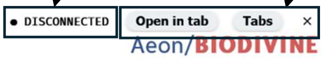
4.1 Aeon Main Page

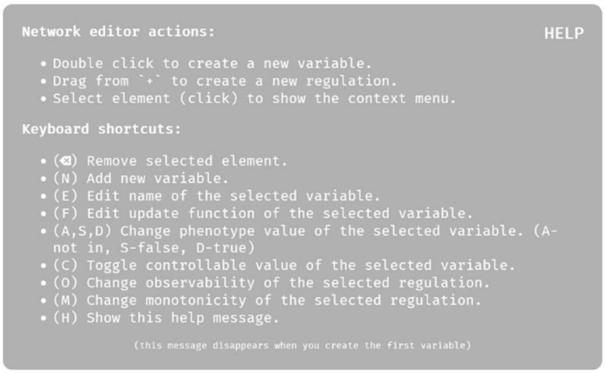


Left panel containing modules for modification of the model and computations

Status bar

Tab panel





Network editor actions:

HELP

- Double click to create a new variable.
- Drag from `+` to create a new regulation.
- Select element (click) to show the context menu.

Keyboard shortcuts:

- (⌫) Remove selected element.
- (N) Add new variable.
- (E) Edit name of the selected variable.
- (F) Edit update function of the selected variable.
- (A,S,D) Change phenotype value of the selected variable. (A-not in, S-false, D-true)
- (C) Toggle controllable value of the selected variable.
- (O) Change observability of the selected regulation.
- (M) Change monotonicity of the selected regulation.
- (H) Show this help message.

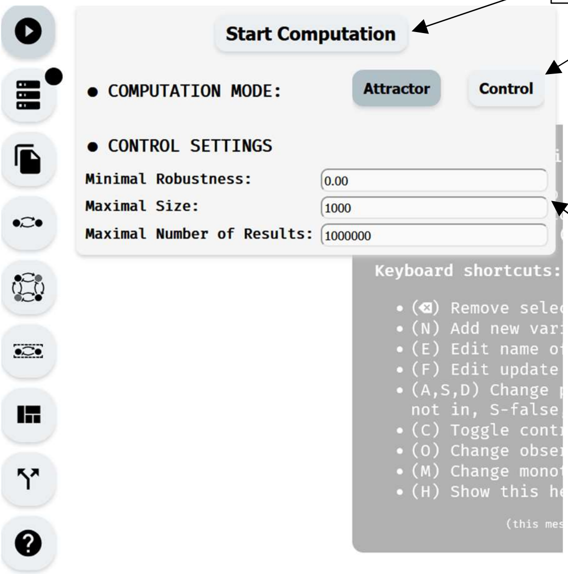
(this message disappears when you create the first variable)

v0.5.0-SNAPSHOT

4.2 Computation module

Button for starting computation

Switch for changing the computation mode



● COMPUTATION MODE: Attractor Control

● CONTROL SETTINGS

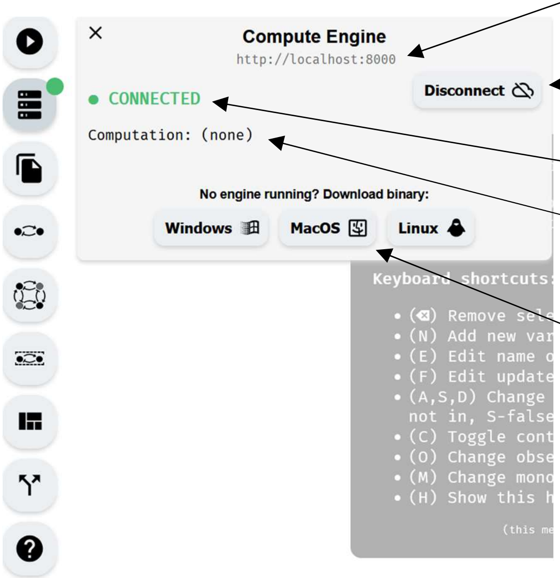
Minimal Robustness: 0.00

Maximal Size: 1000

Maximal Number of Results: 1000000

Settings for parameters of the control computation:
Minimal Robustness (%) = minimal percentage of model parametrizations for which the computed perturbations work
Maximal Size = maximum number of perturbed variables present in computed perturbations
Maximal Number of Results = limits amount of computed results

4.3 Compute Engine module



Address of the running Compute Engine (set by default at http://localhost:8000)

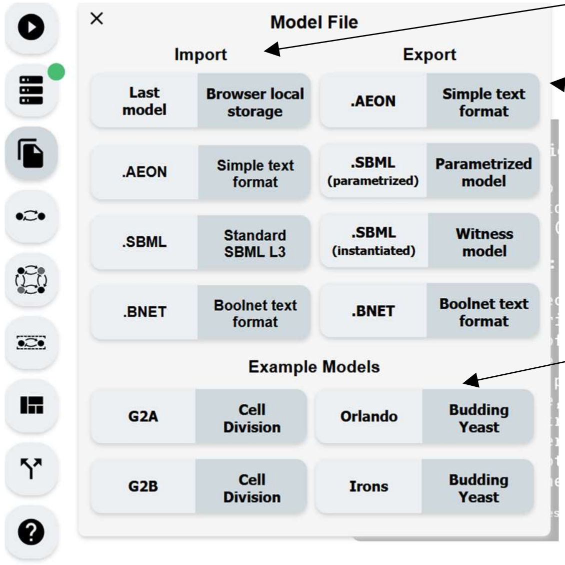
Connect/Disconnect button
To be able to connect Compute Engine needs to run on the same machine

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

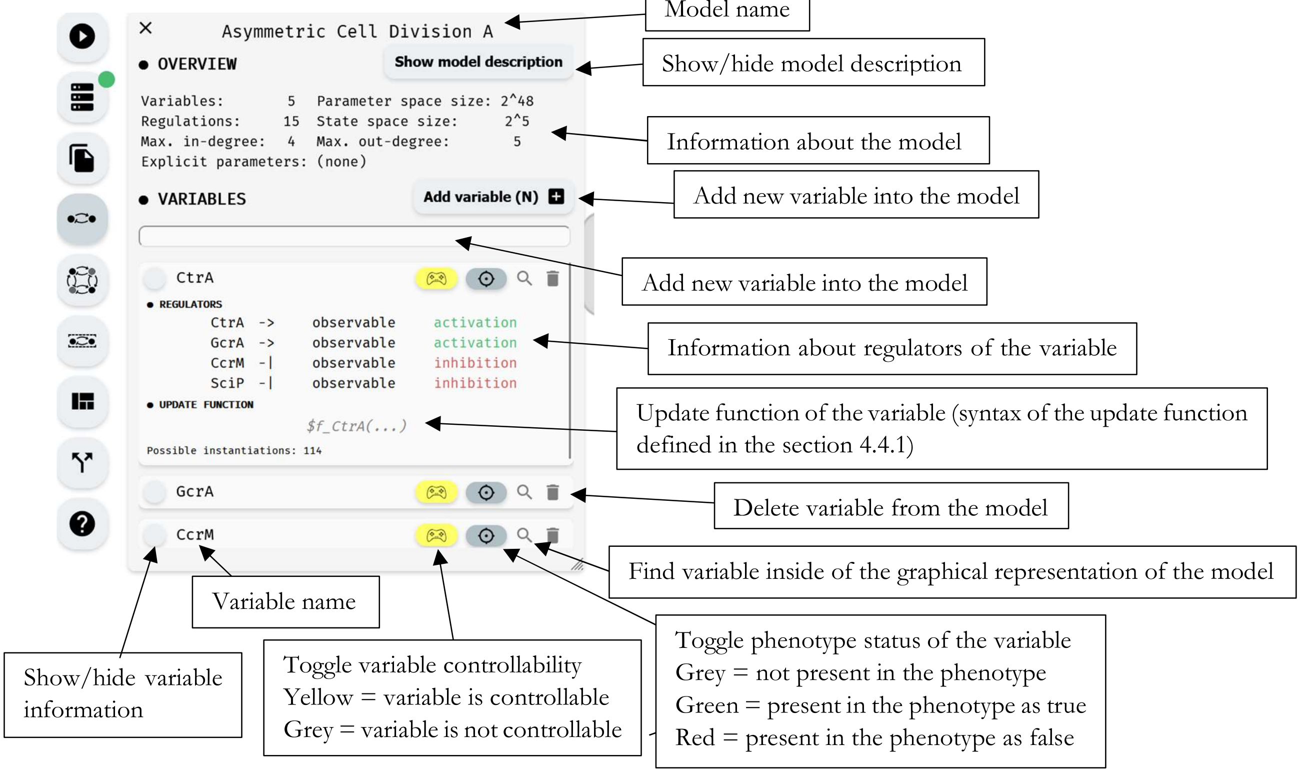


Model import buttons for different formats

Model export buttons for different formats

Import buttons for example models

4.5 Model Editor module



```
×        Asymmetric Cell Division A
● OVERVIEW                      Show model description

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                         Add variable (N)

CtrA
● REGULATORS
      CtrA  ->    observable    activation
      GcrA  ->    observable    activation
      CcrM  -|    observable    inhibition
      SciP  -|    observable    inhibition
● UPDATE FUNCTION
                        $f_CtrA(...)
Possible instantiations: 114

GcrA

CcrM
```

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 4.4.1)

Delete variable from the model

Find variable inside of the graphical representation of the model

Variable name

Show/hide variable information

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

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

4.5.1 Computation Model format and update function syntax

Models are in this format:

Aeon file ::= *Regulation*
| *Update fn decl*
| *Meta*
| *Aeon file*\n*Aeon file*

Update fn decl ::= $ *Name* : *Update fn*

Meta ::= # *Key* : *Value*

Regulation ::= *Name* ␣ *Arrow* ␣ *Name*

Arrow ::= *Kind* | *Kind*?

Kind ::= -> | -| | -?

Update fn ::=
| true | false | *Name* | *Uninterpreted fn*
| ! *Update fn*

Op ::= (*Update fn* *Op* *Update fn*)

Uninterpreted fn ::= & | | | => | <=>

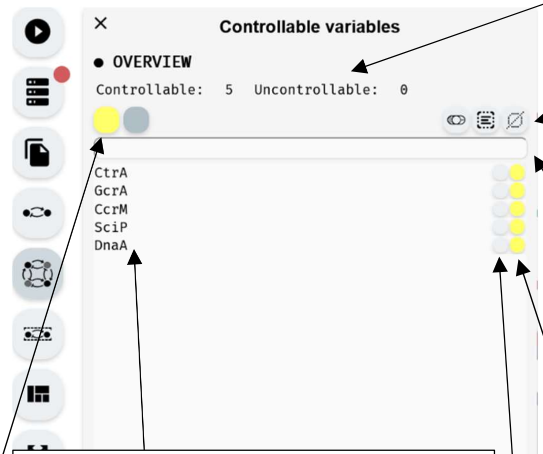
Name (*Parameters*)

Parameters ::= *Name* | *Parameters* , *Parameters*

Update function syntax

Only names of the can be used as function parameters.

4.6 Controllable Editor module



Controllable variables

● OVERVIEW

Controllable: 5 Uncontrollable: 0

CtrA
GcrA
CcrM
SciP
DnaA

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 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)

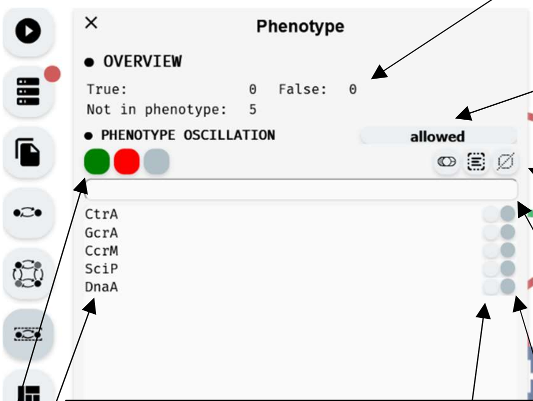
Variable row
Variable can be selected/unselected by clicking/dragging over variable row

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

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

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

4.7 Phenotype Editor module



Phenotype

● OVERVIEW

True: 0 False: 0
Not in phenotype: 5

● PHENOTYPE OSCILLATION allowed

CtrA
GcrA
CcrM
SciP
DnaA

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)

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

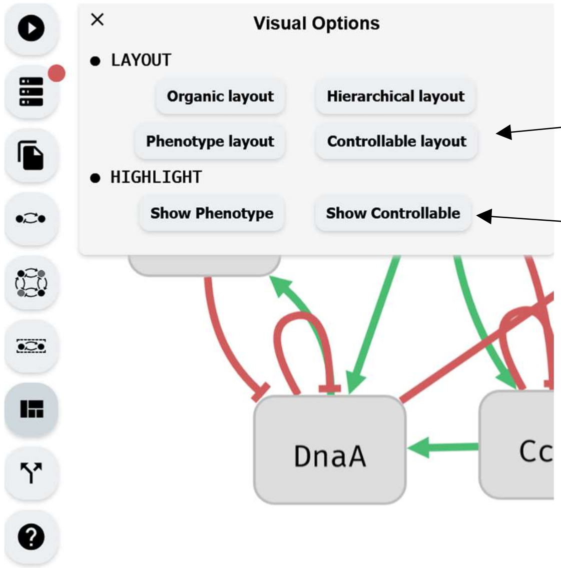
Filter bar
Name of every variable in the filter has to be separated by `,` (example – CtrA, GcrA)
Filter also works for filtering names starting with string (for example if we write `C` into the filter, then table shows all variables starting with C)

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

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

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

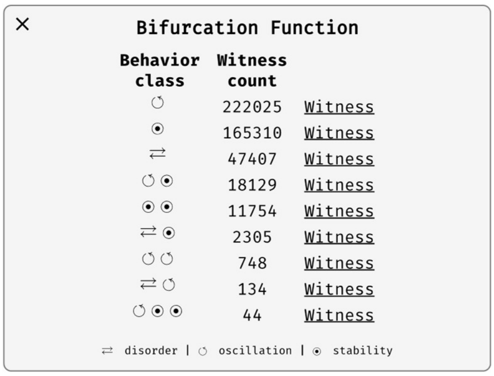
4.8 Visual Options module



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.

Examination of the bifurcation function Result



Bifurcation Function

Behavior class	Witness count	
⟳	222025	Witness
⦿	165310	Witness
⇄	47407	Witness
⟳ ⦿	18129	Witness
⦿ ⦿	11754	Witness
⇄ ⦿	2305	Witness
⟳ ⟳	748	Witness
⇄ ⟳	134	Witness
⟳ ⦿ ⦿	44	Witness

⇄ disorder | ⟳ oscillation | ⦿ stability

Figure 4: An example of a result, representing a bifurcation function

Witness inspection Partition of the parameter space of parametrizations exhibiting the same behaviour

References

[1] NikolaBenešetal.“FormalAnalysisofQualitativeLong-TermBehaviourinParametrised Boolean Networks”. In: *Formal Methods and Software Engineering (ICFEM 2019)*. Springer, 2019, pp. 353–369.

[2] 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.

[3] IsmaelSánchez-Osorio,CarlosA.Hernández-Martínez,andAgustinoMartínez-Antonio. “ModelingAsymmetricCellDivisioninCaulobactercrescentusUsingaBooleanLogic 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.