# Supplementary Information

Riddhiman Bhattacharya

Department of Mathematics, Visva-Bharati University,
Santiniketan, West Bengal 731235, India

**Description**

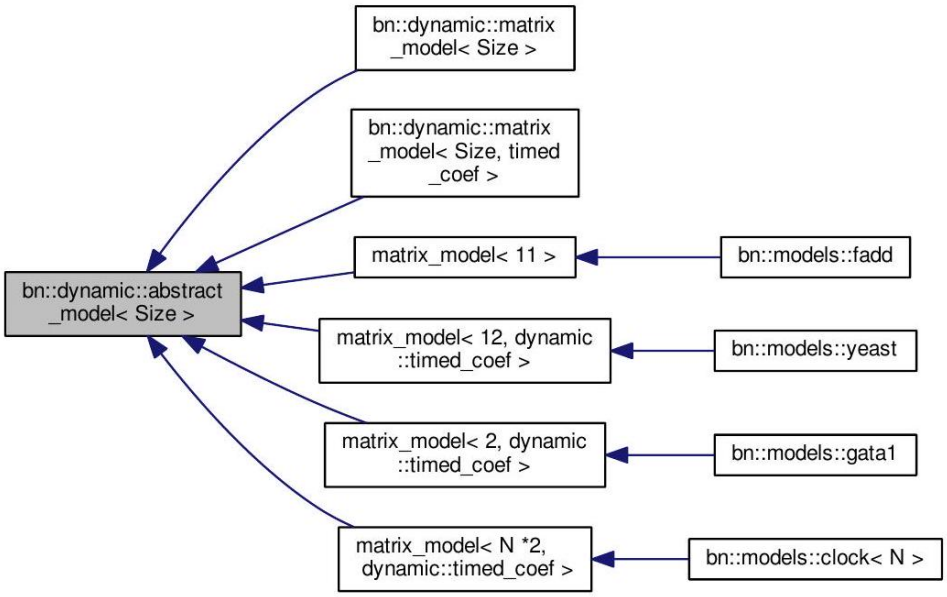
Contains the detailed documentation of **SysBioBoolSim** that contains the code for the implementation of the Boolean network in the paper **A Modular Boolean Automata Framework for Multiscale Simulation of Cell Fate, Cycle, Differentiation, and Circadian Dynamics**.

# Contents

# 1 Class Documentation

## 1.1 bn::dynamic::abstract_model < Size > Class Template Reference

Abstract model of a boolean network.
`#include <abstract_model.h>`
Inheritance diagram for `bn::dynamic::abstract_model < Size >`:




### Public Types

- typedef std::bitset < size > state_type

*Type of container used to store the state of the network.*

### Public Member Functions

- `abstract_model (state_type const &s=state_type())`

*Constructor of the model.*

- state_type const & get_state () const

*Return the current state of the network.*

- virtual std::time_t get_min_time () const = 0

*Return the amount of time a machine can stay static before looping.*

- void set_state (state_type const &s)

*Change the current state.*

- virtual void step ( ) = 0

*Update the model once.*

- bool operator< (`abstract_model` const &other) const

*Comparison between two states of model.*

### Static Public Attributes

- `static std::size_t` const size = Size

*Size of the boolean network.*

### Protected Attributes

- state_type _state

*Current state of the model.*

#### 1.1.1 Detailed Description

**template < std::size_t Size > class bn::dynamic::abstract_model < Size >**
Abstract model of a boolean network.
**Template Parameters**

| | |
|---|---|
| Size | Number of nodes there in the boolean network |

Definition at line 23 of file abstract_model.h.

#### 1.1.2 Constructor & Destructor Documentation

**template<std::size_t Size > bn::dynamic::abstract_model < Size >::abstract_model ( state_type const & *s* =state_type () ) `[inline]`**
Constructor of the model. **Parameters**

| | |
|---|---|
| s | Initial state of the model |

Definition at line 42 of file `abstract_model.h`.

#### 1.1.3 Member Function Documentation

**template < std::size_t Size > virtual std::time_t bn::dynamic::abstract_model < Size >::get_min_time ( ) const `[pure virtual]`**
Return the amount of time a machine can stay static before looping.

### Returns

Minimum time the network has to stay static.
Implemented in bn::dynamic::matrix_model < Size, timed_coef >, bn::dynamic::matrix_model < Size, Coef >, bn::dynamic::matrix_model < 2, dynamic::timed_coef >, bn::dynamic::matrix_model< 11 >, bn::dynamic::matrix_model< 12, dynamic::timed_coef >, bn::dynamic::matrix_model< N * 2, dynamic::timed_coef >, bn::models::clock< N >, bn::models::fadd, bn::models::yeast, and bn::models::gata1.

**template < std::size_t Size > state_type const& bn::dynamic::abstract_model < Size >::get_state ( ) const `[inline]`**
Return the current state of the network.

### Returns

Current state.
Definition at line 51 of file abstract_model.h.

`template < std::size_t Size > bool bn::dynamic::abstract_model < Size >::operator < ( abstract_model < Size > const & other ) const`

`[inline]`

Comparison between two states of model.
Useful if we want to store our model in a BST (binary search tree)
Definition at line 87 of file abstract_model.h.

**template < std::size_t Size > void bn::dynamic::abstract_model < Size >::set_state ( state_type const & *s* )** `[inline]`

Change the current state.
**Parameters**

| | |
|---|---|
| s | New state of the model |

Definition at line 66 of file abstract_model.h.

`template < std::size_t Size > virtual void bn::dynamic::abstract_model < Size >::step ( )`

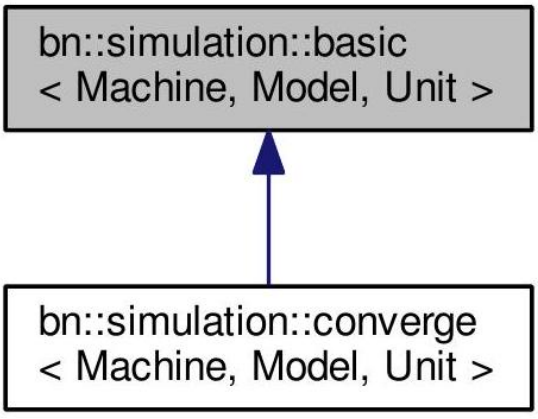
`[pure virtual]` Update the model once.
It's the function used by the machine to update. This function updates a part of the boolean network by following some rules. Each step, the state of the model is calculated again by doing new_state = rule(old_state) where the rule is the transition function.

Implemented in bn::dynamic::matrix_model < Size, timed_coef >, bn::dynamic::matrix_model < Size, Coef >, bn::dynamic::matrix_model < 2, dynamic::timed_coef >, bn::dynamic::matrix_model< 11 >, bn::dynamic::matrix_model < 12, dynamic::timed_coef >, and bn::dynamic::matrix_model < N *2, dynamic::timed_coef >.
The documentation for this class was generated from the following file:

- include/bool_network/dynamic/abstract_model.h

## 1.2 bn::simulation::basic < Machine, Model, Unit > Class Template Reference

Basic simulation of state machine.
`#include <basic.h>`
Inheritance diagram for bn::simulation::basic < Machine, Model, Unit >:




### Public Types

- typedef Unit unit_type

*Type of the time of the simulation.*

### Public Member Functions

- basic (Model const &m=Model())

*Constructor of a simulator.*

- void set_state (typename Model::state_type const &state)

*Setter to modify the state of the model.*

- Model const & get_model () const

*Get the current model. (const version)*

- Model & get_model ()

*Get the current model. (non-const version)*

- unit_type const & get_time () const

*Get the current local time.*

- virtual Model const & advance_from (typename Model::state_type const &state, unit_type const &nbr_step=1)

*Advance the simulation from the given state.*

- virtual Model const & advance (unit_type const &nbr_step=1)

*Advance the simulation by a given number of steps.*

**Protected Attributes**

- Model _model

*Model used by the simulation.*

- unit_type _time

*Local time of the simulation.*

### 1.2.1 Detailed Description

**template<template< typename M > class Machine, typename Model, typename Unit = std::time_t>class bn::simulation::basic < Machine, Model, Unit >**
Basic simulation of state machine.
**Template Parameters**

| | |
|---|---|
| *Machine* | Type of the machine used for the simulation |
| *Model* | Type of the model used |

The machine waits the model as a template argument. For each simulation, one new machine is created and the current model is injected into. The whole memory is so stored in the model. This assures that the machine is only a functional machine which doesn't stock any data.
Definition at line 28 of file basic.h.

### 1.2.2 Member Typedef Documentation

**template<template< typename M > class Machine, typename Model, typename Unit = std::time_t > typedef Unit bn::simulation::basic < Machine, Model, Unit > ::unit_type**
Type of the time of the simulation.
unit_type: It's the type of the unit of time in the simulation. Even if the type is not integral, to let a good integration with the boolean state machine behind, one real step is done only when one integral step is done.
Definition at line 41 of file basic.h.

### 1.2.3 Constructor & Destructor Documentation

**template<template< typename M > class Machine, typename Model, typename Unit = std::time_t> bn::simulation::basic < Machine, Model, Unit >::basic ( Model const & *m* = Model () )** `[inline]`
Constructor of a simulator. **Parameters**

| | |
|---|---|
| *m* | The model to use for the simulation |

Definition at line 47 of file basic.h.

### 1.2.4 Member Function Documentation

**template<template< typename M > class Machine, typename Model, typename Unit = std::time_t> virtual Model const& bn::simulation::basic < Machine, Model, Unit >::advance ( unit_type const & nbr_step = 1 )** `[inline], [virtual]`
Advance the simulation by a given number of steps.
**Parameters**

| | |
|---|---|
| nbr_step | Number of steps the simulation has to advance. |

**Returns**

Model after the simulation.
Creates a new machine and injects a copy of the current model in. Then advance the machine nbr_step times.
Copy the machine's model in the simulation model and return this one.

Reimplemented in bn::simulation::converge< Machine, Model, Unit > .
Definition at line 113 of file basic.h.

```
template < template < typename M > class Machine, typename Model, typename Unit = std::time_t > virtual
Model const& bn::simulation::basic < Machine, Model, Unit >::advance_from ( typename Model::state_type
const & state, unit_type const & nbr_step = 1 )
```

`[inline], [virtual]`

Advance the simulation from the given state.

**Parameters**

| | |
|---|---|
| state | The state to start the simulation from. |
| nbr_step | The number of step the simulation has to advance. |

**Returns**

Return the model after the simulation
Set the state and use the advance function.

**See Also**

advance
Definition at line 96 of file basic.h.

**template<template< typename M > class Machine, typename Model, typename Unit = std::time_t> void bn::simulation::basic < Machine, Model, Unit >::set_state ( typename Model::state_type const & state )** `[inline]`
Setter to modify the state of the model. **Parameters**

| | |
|---|---|
| state | The state to put in the model. |

Definition at line 57 of file basic.h.
The documentation for this class was generated from the following file:

- include/bool_network/simulation/basic.h

## 1.3 bn::abstract_models::clock < Size > Class Template Reference

Model of clock.
`#include <clock.h>`

### Public Member Functions

- virtual bool get_Clk (std::size_t) const = 0

  *Get the state of the clock n.*

- virtual void set_Clk (std::size_t, bool, std::size_t=0)=0

  *Set the state of the clock n.*

- virtual void active_Clk ()=0

  *Active all the clock.*

### Static Public Attributes

- static std::size_t const **size** = Size

### Friends

- std::ostream & operator« (std::ostream &out, abstract_models::clock < Size > const &m)

  *Overloaded operator to show a clock.*

#### 1.3.1 Detailed Description

**template < std::size_t Size > class bn::abstract_models::clock < Size >**
Model of clock.
Lists all the possible interactions with a model of the clock.
Definition at line 23 of file clock.h.

#### 1.3.2 Member Function Documentation

**template<std::size_t Size> virtual void bn::abstract_models::clock< Size >::active_Clk ( )** `[pure virtual]`
Active all the clock.
Use the global input of all clock.
Implemented in bn::models::clock < N >.

#### 1.3.3 Friends And Related Function Documentation

**template < std::size_t Size > std::ostream & operator \ll ( std::ostream & out, abstract_models::clock < Size > const & *m* )** `[friend]`
Overloaded operator to show a clock.
**Parameters**

| | |
|---|---|
| out | The output stream to use |
| *m* | The model to show |

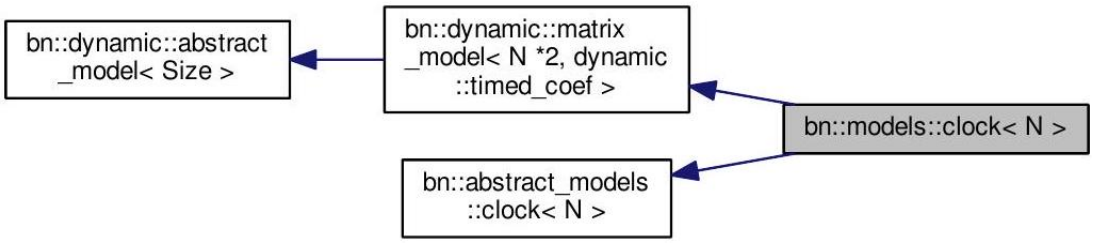
**Returns**

Return the output stream after the operation
Definition at line 50 of file clock.h.
The documentation for this class was generated from the following file:

- include/bool_network/abstract_models/clock.h

## 1.4 bn::models::clock < N > Class Template Reference

Inheritance diagram for bn::models::clock < N > :




### Public Types

- typedef dynamic::matrix_model < size, dynamic::timed_coef > ::state_type **state_type**
- typedef dynamic::matrix_model < size, dynamic::timed_coef > ::matrix_type **matrix_type**
- typedef dynamic::matrix_model < size, dynamic::timed_coef > ::coef_type **coef_type**
- typedef clock_info **clock_info_type** [N]

### Public Member Functions

- **clock** (clock_info_type const &info, state_type const &state=state_type())
- **for** (std::size_t i=0;i< number;i++)
- virtual std::time_t get_min_time () const

*Return the amount of time a machine can stay static before looping.*

- virtual bool get_Clk (std::size_t n) const

  *Get the state of the clock n.*

- virtual void set_Clk (std::size_t n , bool s, std::size_t offset=0)

  *Set the state of the clock n.*

- virtual void active_Clk ()

  *Active all the clock.*

### Public Attributes

- **size**
- **state**

### Static Public Attributes

- static std::size_t const **number** = N
- static std::size_t const **size**

## Additional Inherited Members

### 1.4.1 Detailed Description

`template<std::size_t N>class bn::models::clock<N>`

Definition at line 35 of file clock.h.

### 1.4.2 Member Function Documentation

`template<std::size_t N> virtual void bn::models::clock<N>::active_Clk ( ) [inline], [virtual]`

Active all the clock.
Use the global input of all clock
Implements bn::abstract_models::clock< N >.
Definition at line 129 of file clock.h.

`template < std::size_t N > virtual std::time_t bn::models::clock < N >::get_min_time ( ) const [inline], [virtual]`

Return the amount of time a machine can stay static before looping.

**Returns**

Minimum time the network has to stay static.
Reimplemented from bn::dynamic::matrix_model < N * 2, dynamic::timed_coef >.
Definition at line 89 of file clock.h.

### 1.4.3 Member Data Documentation

`template < std::size_t N > std::size_t const bn::models::clock < N >::size [static]`

**Initial value:**
= dynamic::matrix_model<N * 2, dynamic::timed_coef>::size
Definition at line 41 of file clock.h.

`template < std::size_t N > bn::models::clock < N >::state`

**Initial value:**
= `typedef dynamic::matrix_model<size, dynamic::timed_coef> super`
Definition at line 57 of file clock.h.
The documentation for this class was generated from the following file:

- include/bool_network/models/clock.h

## 1.5 bn::models::clock_info Struct Reference

### Public Member Functions

- **clock_info** (std::time_t t_on=1, std::time_t t_off=1, std::time_t shift=0)

### Public Attributes

- time_t **time_on**
- time_t **time_off**
- time_t **shift**

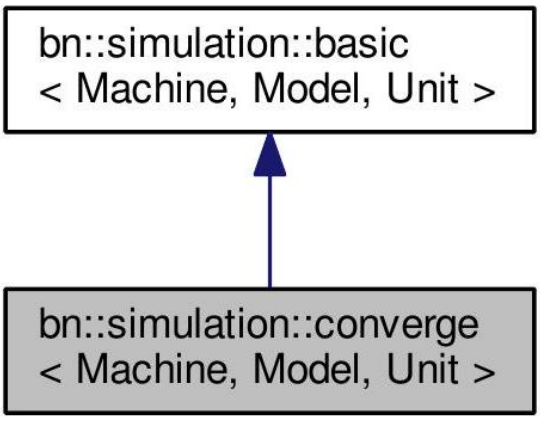
### 1.5.1 Detailed Description

Definition at line 18 of file clock.h.
The documentation for this struct was generated from the following file:

- include/bool_network/models/clock.h

## 1.6 bn::simulation::converge < Machine, Model, Unit > Class Template Reference

Simulation of state machine with keeping a trace of the passage.
`#include <converge.h>`
Inheritance diagram for bn::simulation::converge< Machine, Model, Unit >:




### Public Types

- typedef std::map< Model, std::size_t > visited_type
  *Type of the container of visited states.*
- typedef basic< Machine, Model, Unit >::unit_type unit_type

*Reproduce the parent's type unit_type.*

### Public Member Functions

- converge (Model const &m=Model())
  *Constructor of a simulator.*
- virtual Model const & advance (unit_type const &nbr_step=1)
  *Advance the simulation by a given number of steps.*
- visited_type const & get_visited () const
  *Get the list of visited states.*

**Protected Attributes**

- visited_type_visited
  *The list of visited states during the simulation.*

### 1.6.1 Detailed Description

`template<template< typename M > class Machine, typename Model, typename Unit = std::size_t>class bn::simulat`
`::converge< Machine, Model, Unit >`

Simulation of state machine with keeping a trace of the passage.
**Template Parameters**

| | |
|---|---|
| *Machine* | Type of the machine used for the simulation |
| *Model* | Type of the model used |

The machine waits the model as a template argument. For each simulation, one new machine is created and the current model is injected into. The whole memory is stored in the model. This assures that the machine is only a functional machine which doesn't stock any data.
Definition at line 31 of file converge.h.

### 1.6.2 Member Typedef Documentation

`template<template< typename M > class Machine, typename Model, typename Unit = std::size_t> bn::simulation:`
`Machine, Model, Unit >::unit_type`

*Reproduce the parent's type unit_type.*

**See Also**

basic<Machine, Model, Unit>::unit_type
Definition at line 48 of file converge.h.

`template<template< typename M > class Machine, typename Model, typename Unit = std::size_t> bn::simulation:`
`Machine, Model, Unit >::visited_type`

Type of the container of visited states.
The visited states are stored with their corresponding model allowing the user to used the model's interface.
Definition at line 41 of file converge.h.

### 1.6.3 Constructor & Destructor Documentation

**template<template< typename M > class Machine, typename Model, typename Unit = std::size_t> bn::simulation::converge< Machine, Model, Unit >::converge ( Model const & m = `Model() ` ) [inline]**
Constructor of a simulator.

**Parameters**

| | |
|---|---|
| *m* | The model to use for the simulation |

Stock the given model and initialize the list of converged states.
Definition at line 57 of file converge.h.

### 1.6.4 Member Function Documentation

`template<template< typename M > class Machine, typename Model, typename Unit = std::size_t> virtual`
`Model const& bn::simulation::converge< Machine, Model, Unit >::advance ( unit_type const & nbr_step`
`= 1 ) [inline], [virtual]`

Advance the simulation by a given number of steps.

**Parameters**

| | |
|---|---|
| *nbr_step* | Number of steps the simulation has to advance. |

**Returns**

Model after the simulation.
Creates a new machine and injects a copy of the current model in. Then advance the machine nbr_step times and store the current state in the visited list. Copy the machine's model in the simulation model and return this one.
Reimplemented from bn::simulation::basic< Machine, Model, Unit >.
Definition at line 73 of file converge.h.

```
template<template< typename M > class Machine, typename Model, typename Unit = std::size_t> visited_type
const& bn::simulation::converge< Machine, Model, Unit >::get_visited ( ) const [inline]
```

Get the list of visited states.

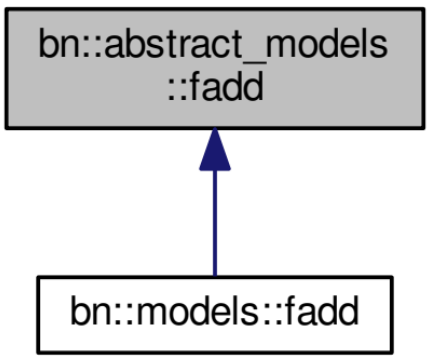
**Returns**

Return the list of visited states.
Definition at line 100 of file converge.h.
The documentation for this class was generated from the following file:

- include/bool_network/simulation/converge.h

## 1.7 bn::abstract_models::fadd Class Reference

Model of FADD.
`#include <fadd.h>` Inheritance diagram for bn::abstract_models::fadd:




**Public Member Functions**

- virtual bool get_TNF () const = 0
  *Get the TNF.*
- virtual bool get_FAS () const = 0
  *Get the FAS.*
- virtual bool get_RIP1 () const = 0
  *Get the RIP1.*
- virtual bool get_NFkB () const = 0
  *Get the NFkB.*
- virtual bool get_C8 () const = 0
  *Get the CASP8.*
- virtual bool get_cIAP () const = 0
  *Get the cIAP.*

- virtual bool get_ATP () const = 0
  *Get the ATP.*
- virtual bool get_C3 () const = 0
  *Get the CASP3.*
- virtual bool get_ROS () const = 0
  *Get the ROS.*
- virtual bool get_MOMP () const = 0
  *Get the MOMP.*
- virtual bool get_MPT () const = 0
  *Get the MPT.*
- virtual void set_TNF (bool) = 0
  *Set the TNF.*
- virtual void set_FAS (bool) = 0
  *Set the FAS.*
- virtual void set_RIP1 (bool) = 0
  *Set the RIP1.*
- virtual void set_NFkB (bool) = 0
  *Set the NFkB.*
- virtual void set_C8 (bool) = 0
  *Set the CASP8.*
- virtual void set_cIAP (bool) = 0
  *Set the cIAP.*
- virtual void set_ATP (bool) = 0
  *Set the ATP.*
- virtual void set_C3 (bool) = 0
  *Set the CASP3.*
- virtual void set_ROS (bool) = 0
  *Set the ROS.*
- virtual void set_MOMP (bool) = 0
  *Set the MOMP.*
- virtual void set_MPT (bool) = 0
  *Set the MPT.*

**Friends**

- std::ostream & operator« (std::ostream &, abstract_models::fadd const &)

*Overloaded operator to show a FADD model.*

### 1.7.1 Detailed Description

Model of FADD.
Lists all the possible interactions with a model of the FADD (Fas-Associated protein with Death Domain).
Definition at line 22 of file fadd.h.

### 1.7.2 Friends And Related Function Documentation

`std::ostream& operator«(std::ostream& out, abstract_models::fadd const & m) [friend]`
Overloaded operator to show a FADD model.

**Parameters**

| | |
|---|---|
| *out* | The output stream to use |
| *m* | The model to show |

**Returns**

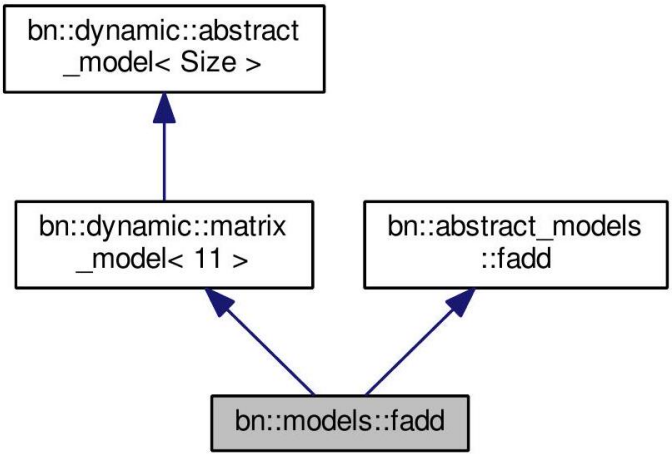
Return the output stream after the operation.

The documentation for this class was generated from the following file:

- `include/bool_network/abstract_models/fadd.h`

## 1.8 bn::models::fadd Class Reference

Inheritance diagram for bn::models::fadd:




**Public Types**

- typedef dynamic::matrix_model < Size >::state_type **state_type**
- typedef dynamic::matrix_model < Size >::matrix_type **matrix_type**
- typedef dynamic::matrix_model < Size >::coef_type **coef_type**

**Public Member Functions**

- **fadd** (std::size_t nbr_updated_node, state_type const &s=state_type())
- virtual std::time_t get_min_time () const

*Return the amount of time a machine can stay static before looping.*

- virtual bool get_TNF () const
  *Get the TNF.*
- virtual bool get_FAS () const
  *Get the FAS.*
- virtual bool get_RIP1 () const
  *Get the RIP1.*
- virtual bool get_NFkB () const
  *Get the NFkB.*
- virtual bool get_C8 () const
  *Get the CASP8.*

- virtual bool get_cIAP () const
  *Get the cIAP.*
- virtual bool get_ATP () const
  *Get the ATP.*
- virtual bool get_C3 () const
  *Get the CASP3.*
- virtual bool get_ROS () const
  *Get the ROS.*
- virtual bool get_MOMP () const
  *Get the MOMP.*
- virtual bool get_MPT () const
  *Get the MPT.*
- virtual void set_TNF (bool)
  *Set the TNF.*
- virtual void set_FAS (bool)
  *Set the FAS.*
- virtual void set_RIP1 (bool)
  *Set the RIP1.*
- virtual void set_NFkB (bool)
  *Set the NFkB.*
- virtual void set_C8 (bool)
  *Set the CASP8.*
- virtual void set_cIAP (bool)
  *Set the cIAP.*
- virtual void set_ATP (bool)
  *Set the ATP.*
- virtual void set_C3 (bool)
  *Set the CASP3.*
- virtual void set_ROS (bool)
  *Set the ROS.*
- virtual void set_MOMP (bool)
  *Set the MOMP.*
- virtual void set_MPT (bool)
  *Set the MPT.*

**Static Public Member Functions**

- static fadd **wild_type** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **anti_oxidant** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **APAF1_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **BAX_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **BCL2_expr** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **C8_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **C8_expr** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **cFlip_del** (std::size_t nbr_updated_node, state_type const &s=state_type())

- static fadd **cIAP_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **FADD_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **NFkB_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **NFkB_expr** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **RIP1_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **XIAP_del** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **z_VAD** (std::size_t nbr_updated_node, state_type const &s=state_type())
- static fadd **z_VAD_RIP1_del** (std::size_t nbr_updated_node, state_type const &s=state_type())

**Static Public Attributes**

- static std::size_t const **Size** = 11
- static std::size_t const **size** = dynamic::matrix_model < Size > ::size

**Protected Member Functions**

- **fadd** (matrix_type const &m, std::size_t nbr_updated_node, state_type const &s=state_type())

**Additional Inherited Members**

### 1.8.1 Detailed Description

Definition at line 14 of file fadd.h.

### 1.8.2 Member Function Documentation

**virtual std::time_t bn::models::fadd::get_min_time ( ) const** `[virtual]`

Return the amount of time a machine can stay static before looping.

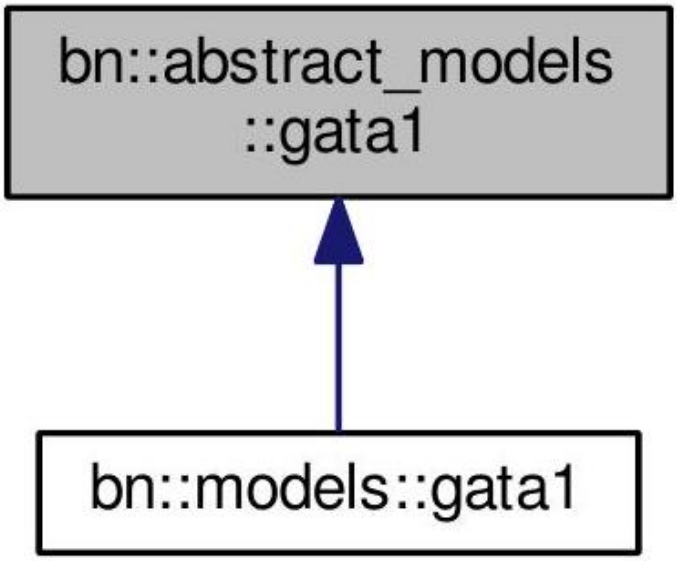
**Returns**

Minimum time the network has to stay static.
Reimplemented from bn::dynamic::matrix_model ⟨11⟩.
The documentation for this class was generated from the following file:

- include/bool_network/models/fadd.h

## 1.9 bn::abstract_models::gata1 Class Reference

Model representing the activation of GATA-1 by EPO.
`#include <gata1.h>` Inheritance diagram for bn::abstract_models::gata1:




**Public Member Functions**

- virtual bool get_Epo () const = 0

  *Get the Epo (erytropoietin).*

- virtual bool get_GATA1 () const =0

  *Get the GATA-1.*

- virtual void set_Epo (bool)=0

  *Set the Epo (erytropoietin)*

- virtual void set_GATA1 (bool)=0

  *Set the GATA-1.*

**Friends**

- std::ostream & operator« (std::ostream &, abstract_models::gata1 const &)

  *Overloaded operator to show a GATA-1 model.*

### 1.9.1 Detailed Description

Model representing the activation of GATA-1 by Epo.
Lists all the possible interactions with a model representing the activation of GATA-1 by a certain amount of Epo.
Definition at line 22 of file gata1.h.

### 1.9.2 Friends And Related Function Documentation

**std::ostream & operator « ll ( std::ostream & , abstract_models::gata1 const & )** `[friend]`
Overloaded operator to show a GATA-1 model.

**Parameters**

| | |
|---|---|
| out | The output stream to use |
| *m* | The model to show |

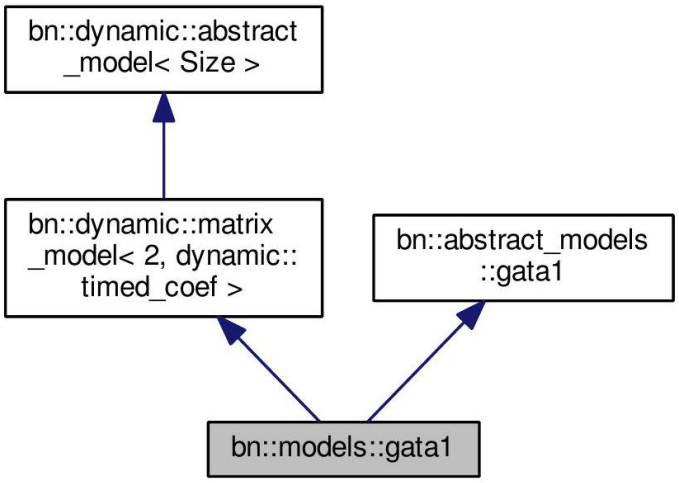
**Returns**

Return the output stream after the operation.
The documentation for this class was generated from the following file:

- include/bool_network/abstract_models/gata1.h

## 1.10 bn::models::gata1 Class Reference

Inheritance diagram for bn::models::gata1:




### Public Types

- typedef dynamic::matrix_model < Size, dynamic::timed_coef > ::state_type **state_type**
- typedef dynamic::matrix_model < Size, dynamic::timed_coef > ::matrix_type **matrix_type**
- typedef dynamic::matrix_model < Size, dynamic::timed_coef > ::coef_type **coef_type**

### Public Member Functions

- **gata1** (std::time_t const &td, state_type const &s=state_type())
- virtual std::time_t get_min_time () const

  *Return the amount of time a machine can stay static before looping.*

- virtual bool get_Epo () const

  *Get the Epo.*

- virtual bool get_GATA1 () const

  *Get the GATA1.*

- virtual void set_Epo (bool a)

  *Set the Epo.*

- virtual void set_GATA1 (bool a)

  *Set the GATA1.*

### Static Public Attributes

- static std::size_t const **Size** = 2
- static std::size_t const **size** = dynamic::matrix_model <Size > ::size

### Protected Attributes

- std::time_t **_td**

### Additional Inherited Members

#### 1.10.1 Detailed Description

Definition at line 14 of file gata1.h.

#### 1.10.2 Member Function Documentation

`virtual std::time_t bn::models::gata1::get_min_time ( ) const [virtual]`

Return the amount of time a machine can stay static before looping.

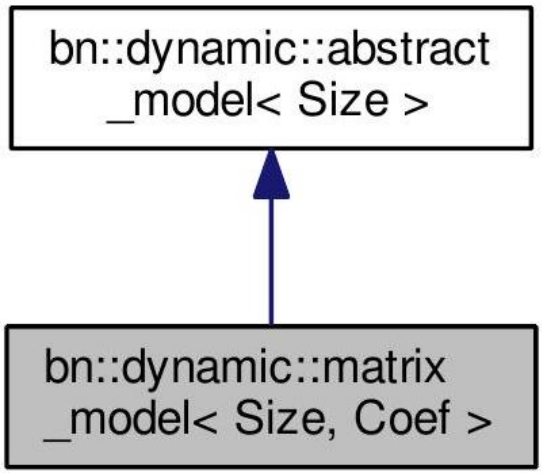
### Returns

Minimum time the network has to stay static.
Reimplemented from bn::dynamic::matrix_model < 2, dynamic::timed_coef >.
The documentation for this class was generated from the following file:

- include/bool_network/models/gata1.h

## 1.11 bn::dynamic::matrix_model < Size, Coef > Class Template Reference

Model of a boolean network based on a matrix of transition.
`#include <matrix_model.h>`
Inheritance diagram for bn::dynamic::matrix_model < Size, Coef > :




### Public Types

- enum update {**activation, stase, deactivation** }

  Enumeration of the possible modification on one node's machine.

- typedef abstract_model < Size >
  ::state_type **state_type**
  *Type of a state of the machine.*

- typedef Coef coef_type

  *Type contained by the matrix of transition.*

- typedef coef_type matrix_type [(size+1)*size]

  *Type of a matrix of transition.*

### Public Member Functions

- matrix_model (matrix_type const &m, std::size_t nbr_updated_node, state_type const &s=state_type())

*Constructor of the model.*

- virtual std::time_t get_min_time () const

*Return the amount of time a machine can stay static before looping.*

- virtual void step ()

*Update the model once.*

- virtual void pick_modification (state_type const &s, std::size_t list_modified[size], std::size_t size_modified)

*Pick some modification of the potential future state to the current state.*

### Static Public Attributes

- static std::size_t const size = abstract_model <Size > ::size

*Size of the Boolean network.*

### Protected Member Functions

- virtual update rule (std::size_t n)

*Get the modification of the given node.*

## Protected Attributes

- matrix_type _matrix

*Matrix of transition.*

- std::size_t_nbr_updated_node

*Amount of node updated each step.*

### 1.11.1 Detailed Description

**template < std::size_t Size, typename Coef = float > class bn::dynamic::matrix_model < Size, Coef >**

Model of a boolean network based on a matrix of transition.

### Template Parameters

| | |
|---|---|
| Size | The size of the Boolean network. |
| Coef | The type of coefficient contained in the matrix. The transition rules are stored in a matrix similar to a Markov chain. A node can be connected to another node through a coefficient. When a node is updated, the sum of all coefficients of the active nodes connected to it is computed. If the sum is greater than a threshold, the node is activated; if it is lower, the node is deactivated; and if it is zero, the node remains unchanged. |

Definition at line 34 of file matrix_model.h.

### 1.11.2 Member Typedef Documentation

**template < std::size_t Size, typename Coef = float > bn::dynamic::matrix_model < Size, Coef >::matrix_type**

Type of a matrix of transition.
It's the type of the matrix. The size is (size +1 ) ∗ size because there is a line for the threashold information.
Definition at line 62 of file matrix_model.h.

## Constructor & Destructor Documentation

**template<std::size_t Size, typename Coef = float > bn::dynamic::matrix_model < Size, Coef >::matrix_model ( matrix_type const & *m*, std::size_t *nbr_updated_node*, state_type const & *s* = state_type ( ) ) [inline]**

Constructor of the model.

### Parameters

| | |
|---|---|
| *m* | Matrix of transition used |
| *nbr_updated_node* node | Number of node updated each step |
| *s* | Initial state of the model |

It's possible to have two kinds of different models. Each step, it may have more than one node to modify. So there is some different way to update the model. The first, is to choose all the modification. Each time every node are modified (if there is a modification). It's called a synchronous update. Another model can describe the same network but with a different method for choosing a node. If only one node is chosen randomly, then from one state, there are many other states. This kind of model is also non-deterministic and it's called asynchronous. It's possible to get a middle of async and sync by updating a certain amount of node. If this amount is greater than the network size, so the model is sync and deterministic.
Definition at line 90 of file matrix_model.h.

### Member Function Documentation

**template < std::size_t Size, typename Coef = float > virtual std::time_t bn::dynamic::matrix_model < Size, Coef >::get_min_time( )const [inline], [virtual]**

Return the amount of time a machine can stay static before looping.

### Returns

Minimum time the network has to stay static.
Implements bn::dynamic::abstract_model < Size >.
Reimplemented in bn::models::clock< N >, bn::models::fadd, bn::models::yeast, and bn::models::gata1.
Definition at line 104 of file matrix_model.h.

**template<std::size_t Size, typename Coef = float> virtual void bn::dynamic::matrix_model< Size, Coef >::pick_modification ( state_type const & s, std::size_t list_modified[size], std::size_t size_modified ) [inline], [virtual]**

Pick some modification of the potential future state to the current state.

### Parameters

| | |
|---|---|
| *s* | Potential future state containing all the modification |
| list_modified | List of th index of all the modification done |
| size_modified | Size of the list |

Get _nbr_updated_node times modification in the potential future state to set it in the current state to get the new one. The choice is random, so if the number of modification taken is lower than the number of nodes, the result is not deterministic.

## See Also

_nbr_updated_node

## Warning

The current random generator is std::rand from cstdlib.
Definition at line 160 of file matrix_model.h.

#### 4.11.4.3 template < std::size_t Size, typename Coef = float > virtual update bn::dynamic::matrix < Size, Coef >::rule (std::size_tn) [inline],[protected],[virtual]

Get the modification of the given node.

## Returns

Type of modification

Do the sum of all coefficient of active node attached to those given. If the result is strictly positive, it's an activation, if it's strictly negative, it's a deactivation. Else the node stays the same.

Definition at line 194 of file matrix_model.h.

#### 4.11.4.4 template < std::size_t Size, typename Coef = float > virtual void bn::dynamic::matrix_n < Size, Coef >::step ( ) [inline],[virtual]

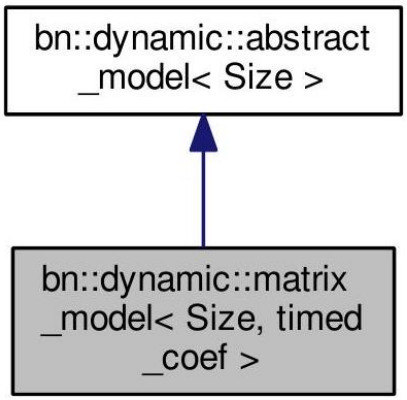
Update the model once.
Uses the sum rule on each node to determine the potential future state. Then picks some modification from the potential future state and include them in the current state to get the new.

Implements bn::dynamic::abstract_model< Size >.
Definition at line 116 of file matrix_model.h.
The documentation for this class was generated from the following file:

- include/bool_network/dynamic/matrix_model.h

### 1.12 bn::dynamic::matrix_model < Size, timed_coef > Class Template Reference

Inheritance diagram for bn::dynamic::matrix_model < Size, timed_coef > :




**Public Types**

- enum update {**activation, stase, deactivation** }

*Enumeration of the possible modification on one node's machine.*

- typedef abstract_model < Size > ::state_type state_type

*The type of a state of the machine.*

- typedef timed_coef coef_type

  *The type contained by the matrix of transition.*

- typedef coef_type matrix_type [(size+1)*size]

  *The type of a matrix of transition.*

**Public Member Functions**

- matrix_model (matrix_type const &m, std::size_t nbr_updated_node, state_type const &s=state_type())

  Constructor of the model.

- virtual void step ()

  *Update the model once.*

- virtual void pick_modification (state_type const &s, std::size_t list_modified[size], std::size_t size_modified)

  *Pick some modification from the potential future state to the current state.*

- virtual std::time_t get_min_time () const

  *Returns the amount of time a machine can stay static before looping.*

**Static Public Attributes**

- static std::size_t const size = abstract_model <Size >:: size
  *The size of the boolean network.*

**Protected Member Functions**

- virtual update rule (std::size_t n) *Get the modification of the given node.*

**Protected Attributes**

- matrix_type _matrix
  *The transition matrix.*
- std::size_t _nbr_updated_node
  *The number of nodes updated at each step.*

### 1.12.1 Detailed Description

`template<std::size_t Size>class bn::dynamic::matrix_model< Size, timed_coef >`

Definition at line 72 of file timed_matrix_model.h.

### 1.12.2 Constructor & Destructor Documentation

**template<std::size_t Size> bn::dynamic::matrix_model< Size, timed_coef >::matrix_model ( matrix_type const & m, std::size_t nbr_updated_node, state_type const & s = state_type() ) [inline]**

Constructor of the model.

**Parameters**

| | |
|---|---|
| *m* | The transition matrix used. |
| *nbr_updated_node* | The number of node updated each step. |
| *s* | The initial state of the model. |

It's possible to have two kinds of different models. Each step, it may have more than one node to modify. So there is some different way to update the model. The first, is to choose all the modification. Each time every node are modified (if there is a modification). It's called a synchronous update. Another model can describe the same network but with a different method for choosing a node. If only one node is chosen randomly, then from one state, there are many other states. This kind of model is also non-deterministic and it's called asynchronous. It's possible to get a middle of async and sync by updating a certain amount of node. If this amount is greater than the network size, so the model is sync and deterministic.
Definition at line 125 of file timed_matrix_model.h.

### 1.12.3 Member Function Documentation

**template<std::size_t Size> virtual std::time_t bn::dynamic::matrix_model< Size, timed_coef >::get_min_time ( ) const** `[inline], [virtual]`

Returns the amount of time a machine can stay static before looping.

#### Returns

The minimum time the network has to stay static.
Implements bn::dynamic::abstract_model< Size >.
Definition at line 212 of file timed_matrix_model.h.

**template<std::size_t Size> virtual void bn::dynamic::matrix_model< Size, timed_coef >::pick_modification ( state_type const & s, std::size_t list_modified[size], std::size_t size_modified )** `[inline], [virtual]`

Pick some modification from the potential future state to the current state.

#### Parameters

| | |
|---|---|
| *s* | The potential future state containing all the modifications. |
| *list_modified* | List of the indices of all modifications performed. |
| *size_modified* | The size of the modification list. |

Get _nbr_updated_node times modification in the potential future state to set it in the current state to get the new one. The choice is random, so if the number of modification taken is lower than the number of nodes, the result is not deterministic.

#### See Also

_nbr_updated_node

#### Warning

The current random generator is std::rand from cstdlib.
Definition at line 186 of file timed_matrix_model.h.

**template < std::size_t Size > virtual update bn::dynamic::matrix_model < Size, timed_coef >::rule ( std::size_t *n* )** `[inline], [protected], [virtual]`

Get the modification of the given node.

#### Returns

Return the type of modification.
Do the sum of all coefficient of active node attached to those given. If the result is strictly positive, it's an activation, if it's strictly negative, it's a deactivation. Else the node stays the same.
Definition at line 235 of file timed_matrix_model.h.

```
template<std::size_t Size> virtual void bn::dynamic::matrix_model< Size, timed_coef >::step ( )
   [inline],[virtual]
```

Update the model once.
Uses the sum rule on each node to determine the potential future state. Then picks some modification from the potential future state and include them in the current state to get the new.
Implements bn::dynamic::abstract_model< Size >.
Definition at line 142 of file timed_matrix_model.h.
The documentation for this class was generated from the following file:

- include/bool_network/dynamic/timed_matrix_model.h

## 1.13 bn::dynamic::state_machine < Model > Class Template Reference

State machine.
`#include <state_machine.h>`

### Public Types

- typedef Model **model_type**
- typedef model_type::state_type state_type
  *Represent one state of the machine.*
- typedef std::vector< state_type > history_type
  *Container of the visited state of the machine.*

### Public Member Functions

- state_machine (model_type const &m = model_type())
  *Constructor of the state machine.*
- model_type const & get_model () const
  *Return the current model used.*
- model_type & get_model ()
  *Return the current model used.*
- void step (std::time_t time = 1)
  *Update the machine.*

### Protected Attributes

- model_type _model
  *Current model used.*
- history_type _history
  *List of all visited states.*
- bool _in_cycle
  *Indicates whether the machine is looping.*
- std::size_t _begin_cycle
  *Starting point of the machine's loop.*
- std::time_t _time
  *Local time of the machine.*

### 1.13.1 Detailed Description

`template<typename Model>class bn::dynamic::state_machine< Model>`

State machine.
Template Parameters

| | |
|---|---|
| Model | model used by the machine It's the representation of a state machine defined by the model Model. This machine manages loops in the state graph. In case of a loop, the final step is automatically deduced without calculating those between. All types of model which have a step function work with this network. |

The model has to give some information :

- The type of one state which must have an equal operator.
- The update of the model such as for each step there is new_state = update(old_state)
- The amount of time the machine can stay in stase without considering it's looping or converging

Definition at line 35 of file state_machine.h.

### 1.13.2 Member Typedef Documentation

`template<typename Model>bn::dynamic::state_machine<Model>::history_type`

Container of the visited state of the machine.
For each step, the current state is stored. This prevents for loop in the state graph.
Definition at line 59 of file state_machine.h.

`template < typename Model > bn::dynamic::state_machine < Model >::state_type`

Represent one state of the machine.

**Warning**

The model has to give an equal operator for the state.
The model gives the type of the state. For preventing an infinite loop, all visited states are stored and for each step, the new state is, searched in the visited. This can be used only if there is an operator to check if two states are equal.
Definition at line 50 of file state_machine.h.

### 1.13.3 Constructor & Destructor Documentation

**template<typename Model>bn::dynamic::state_machine< Model >::state_machine ( model_type const & *m* = model_type() ) [inline]**

Constructor of the state machine.

**Parameters**

| | |
|---|---|
| *m* | copy the model given. |

Constructs a state machine by copying the given model. The constructor set also the variables to prevent looping.
The current state of the model is stored in the list of visited state.
Definition at line 69 of file state_machine.h.

### 1.13.4 Member Function Documentation

**template < typename Model > model_type const& bn::dynamic::state_machine < Model >::get_model ( ) const [inline]**

Return the current model used.

**Returns**

The current model.
Definition at line 84 of file state_machine.h.

`template<typename Model> model_type& bn::dynamic::state_machine< Model >::get_model ( ) [inline]`

Return the current model used.

**Returns**

The current model
It's the non-const version.
Definition at line 95 of file state_machine.h.

`4.13.4.3 template<typename Model> void bn::dynamic::state_machine< Model >::step ( std::time_t time = 1 ) [inline]`

Update the machine.

**Parameters**

| | |
|---|---|
| *time* | the number of time the machine is updated. |

Updates the machine and the model. Also detects if there is a loop. In this case, jump directly to the final state. A model can specify the time it can be static without deduce there is a convergence or a loop.
Definition at line 109 of file state_machine.h.

### 1.13.5 Member Data Documentation

`template<typename Model>bn::dynamic::state\_machine< Model >::\_begin\_cycle [protected]`

The begin of the loop of the machine.

**Warning**

If the machine is not in loop, the value may be invalid.
Definition at line 199 of file state_machine.h.
The documentation for this class was generated from the following file:

- include/bool_network/dynamic/state_machine.h

## 1.14 bn::dynamic::timed_coef Struct Reference

Matrix's coefficient with a time retard on the effect.

`\#include <timed\_matrix\_model.h>`

**Public Types**

- typedef float float_type
  *Floating-point type used for the* `time_coef`.

**Public Member Functions**

- timed_coef (float_type coef=0, std::time_t time_min=0, bool reset_time=true, std::time_t time=0)
  *Constructor of a timed coefficient.*

**Public Attributes**

- float_type coef

  Coefficient.

- std::time_t time

  Local time of the effect.

- std::time_t time_min

  Restard's time of the effect.

- bool reset_time

  Indicates is the model has to reset the time after an effect done.

### 1.14.1 Detailed Description

Matrix's coefficient with a time retard on the effect.
Definition at line 23 of file timed_matrix_model.h.

### 1.14.2 Constructor & Destructor Documentation

**bn::dynamic::timed_coef::timed_coef ( float_type coef = 0, std::time_t time_min = 0, bool reset_time = t rue, std::time_t time = 0 )** `[inline]`

Constructor of a timed coefficient.

**Parameters**

| | |
|---|---|
| *coef* | Coefficient |
| *time_min* | The retard the effect has |
| *reset_time* | Indicates if the model has to clear on not the timer after one effect done. |
| *time* | Offset at beginning |

Definition at line 38 of file timed_matrix_model.h.
The documentation for this struct was generated from the following file:

- include/bool_network/dynamic/timed_matrix_model.h

## 1.15 bn::dynamic::timed_matrix_model < Size > Struct Template Reference

Timed model of a boolean network.
`#include <timed_matrix_model.h>`

**Public Types**

- typedef matrix_model < Size, timed_coef > type
  *Shortcut for hide the structure timed_coef.*

### 1.15.1 Detailed Description

**template < std::size_t Size > struct bn::dynamic::timed_matrix_model < Size >**

Timed model of a boolean network.

**Template Parameters**

| | |
|---|---|
| Size | The size of the Boolean network. Transition rules are represented in a matrix similar to a Markov chain. Each coefficient in the matrix denotes a possible connection between nodes. When a node is updated, the sum of coefficients from all active nodes connected to it is computed. If the sum is positive, the node becomes activated; if negative, it is deactivated; and if zero, its state remains unchanged. A node can influence another only if the corresponding matrix time is greater than the defined minimum time. |

Definition at line 305 of file timed_matrix_model.h.

### 1.15.2 Member Typedef Documentation

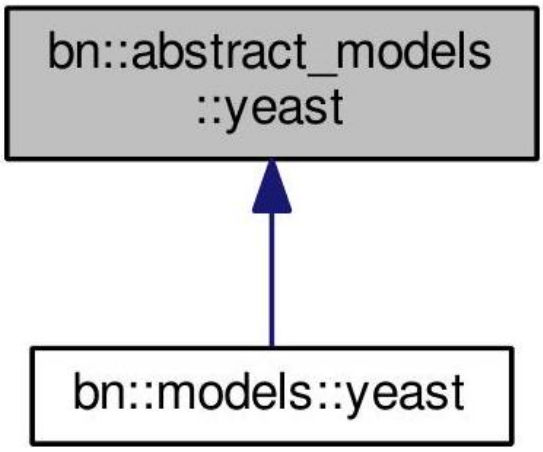
**template < std::size_t Size > bn::dynamic::timed_matrix_model < Size >::type**

Shortcut for hide the structure timed_coef.
It's the type of a matrix_model class using the structure timed_coef as coefficient of the matrix.
Definition at line 314 of file timed_matrix_model.h.
The documentation for this structure was generated from the following file:

- include/bool_network/dynamic/timed_matrix_model.h

## 1.16 bn::abstract_models::yeast Class Reference

Yeast model.
`#include <yeast.h>`
Inheritance diagram for bn:::abstract_models::yeast:




**Public Member Functions**

- virtual bool get_Cell_size () const = 0
  *Get the Cell Size checkpoint.*
- virtual bool get_Cln3 () const = 0
  *Get the Cln3.*
- virtual bool get_SBF () const = 0
  *Get the SBF.*
- virtual bool get_Cln1_2 () const = 0
  *Get the C/n1,2.*
- virtual bool get_Cdh1 () const = 0
  *Get the Cdh1.*
- virtual bool get_Cdc20_Cdc14 () const = 0
  *Get the Cdc20&Cdc14.*
- virtual bool get_Swi5 () const = 0
  *Get the Swi5.*
- virtual bool get_Mcm1_SFF () const = 0
  *Get the Mcm1/SFF.*
- virtual bool get_Clb5_6 () const = 0
  *Get the Clb5,6.*
- virtual bool get_MBF () const = 0
  *Get the MBF.*
- virtual bool get_Sic1 () const = 0
  *Get the Sic1.*
- virtual bool get_Clb1_2 () const = 0
  *Get the Clb1,2.*
- virtual void set_Cell_size (bool) = 0
  *Set the Cell Size checkpoint.*
- virtual void set_Cln3 (bool) = 0
  *Set the Cell Size checkpoint.*
- virtual void set_SBF (bool) = 0
  *Set the SBF.*
- virtual void set_Cln1_2 (bool) = 0
  *Set the C/n1,2.*

- virtual void set_Cdh1 (bool) = 0
  *Set the Cdh1.*
- virtual void set_Cdc20_Cdc14 (bool) = 0
  *Set the Cdc20&Cdc14.*
- virtual void set_Swi5 (bool) = 0
  *Set the Swi5.*
- virtual void set_Mcm1_SFF (bool) = 0
  *Set the Mcm1/SFF.*
- virtual void set_Clb5_6 (bool) = 0
  *Set the Clb5,6.*
- virtual void set_MBF (bool) = 0
  *Set the MBF.*
- virtual void set_Sic1 (bool) = 0
  *Set the Sic1.*
- virtual void set_Clb1_2 (bool) = 0
  *Set the Clb1,2.*

**Friends**

- std::ostream & operator« (std::ostream &, abstract_models::yeast const &)
  *Overloaded operator to show a yeast model.*

### 1.16.1 Detailed Description

Yeast model.
Lists all the possible interactions with a model representing the life of a yeast.
Definition at line 22 of file yeast.h.

### 1.16.2 Friends And Related Function Documentation

**std::ostream& operator« (std::ostream & , abstract_models::yeast const & ) [friend]**

Overloaded operator to show a yeast model.

**Parameters**

| | |
|---|---|
| *out* | The output stream to use |
| *m* | The model to show |

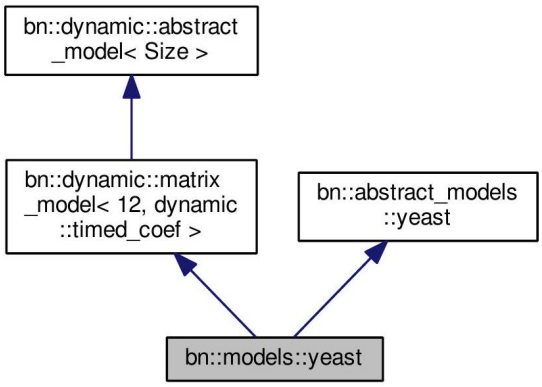
**Returns**

Return the output stream after the operation.
The documentation for this class was generated from the following file:

- include/bool_network/abstract_models/yeast.h

## bn::models::yeast Class Reference

Inheritance diagram for bn::models::yeast:




### Public Types

- typedef dynamic::matrix_model < Size, dynamic::timed_coef > ::state_type **state_type**
- typedef dynamic::matrix_model < Size, dynamic::timed_coef > ::matrix_type **matrix_type**
- typedef dynamic::matrix_model < Size, dynamic::timed_coef > ::coef_type **coef_type**

### Public Member Functions

- **yeast** (std::size_t nbr_updated_node, coef_type const &ag, coef_type const &ar, std::time_t const &td, state_type const &s=state_type())
- virtual std::time_t get_min_time () const
  *Return the amount of time a machine can stay static before looping.*
- virtual bool get_Cell_size () const
  *Get the Cell Size checkpoint.*
- virtual bool get_Cln3 () const
  *Get the CIn3.*
- virtual bool get_SBF () const
  *Get the SBF.*
- virtual bool get_Cln1_2 () const
  *Get the C/n1,2.*
- virtual bool get_Cdh1 () const
  *Get the Cdh1.*
- virtual bool get_Cdc20_Cdc14 () const
  *Get the Cdc20 & Cdc14.*
- virtual bool get_Swi5 () const
  *Get the Swi5.*
- virtual bool get_Mcm1_SFF () const
  *Get the Mcm1/SFF.*
- virtual bool get_Clb5_6 () const
  *Get the Clb5,6.*
- virtual bool get_MBF () const
  *Get the MBF.*

- virtual bool get_Sic1 () const
  *Get the Sic1.*
- virtual bool get_Clb1_2 () const
  *Get the Clb1,2.*
- virtual void set_Cell_size (bool a)
  *Set the Cell Size checkpoint.*
- virtual void set_Cln3 (bool a)
  *Set the Cell Size checkpoint.*
- virtual void set_SBF (bool a)
  *Set the SBF.*
- virtual void set_Cln1_2 (bool a)
  *Set the C/n1,2.*
- virtual void set_Cdh1 (bool a)
  *Set the Cdh1.*
- virtual void set_Cdc20_Cdc14 (bool a)
  *Set the Cdc20 & Cdc14.*
- virtual void set_Swi5 (bool a)
  *Set the Swi5.*
- virtual void set_Mcm1_SFF (bool a)
  *Set the Mcm1/SFF.*
- virtual void set_Clb5_6 (bool a)
  *Set the Clb5,6.*
- virtual void set_MBF (bool a)
  *Set the MBF.*
- virtual void set_Sic1 (bool a)
  *Set the Sic1.*
- virtual void set_Clb1_2 (bool a)
  *Set the Clb1,2.*

**Static Public Attributes**

- static std::size_t const **Size** = 12
- static std::size_t const **size** = dynamic::matrix_model < Size > ::size

**Protected Attributes**

- std::time_t **_td**

**Additional Inherited Members**

### 1.16.3 Detailed Description

Definition at line 14 of file yeast.h.

### 1.16.4 Member Function Documentation

**virtual std::time_t bn::models::yeast::get_min_time ( ) const** `[virtual]`

Return the amount of time a machine can stay static before looping.

**Returns**

Minimum time the network has to stay static.
Reimplemented from bn::dynamic::matrix_model < 12, dynamic::timed_coef >.
The documentation for this class was generated from the following file:

- include/bool_network/models/yeast.h

# 2 File Documentation

## 2.1 include/bool_network/abstract_models/clock.h File Reference

Definition of an interface representing a clock.
`#include <ostream>`

### Classes

- class bn::abstract_models::clock< Size >
  *Model of clock.*

### 2.1.1 Detailed Description

Definition of an interface representing a clock.
Definition in file clock.h.

## 2.2 include/bool_network/abstract_models/fadd.h File Reference

Definition of an interface representing a model of FADD.
`#include <ostream>`

### Classes

- class bn::abstract_models::fadd
  *Model of FADD.*

### 2.2.1 Detailed Description

Definition of an interface representing a model of FADD.
Definition in file fadd.h.

## 2.3 include/bool_network/abstract_models/gata1.h File Reference

Definition of an interface of a model representing the activation of GATA-1 by Epo.
`#include <ostream>`

### Classes

- class bn::abstract_models::gata1
  *Model representing the activation of GATA-1 by Epo.*

### 2.3.1 Detailed Description

Definition of an interface of a model representing the activation of GATA-1 by EPO.
Definition in file gata1.h.

## 2.4 include/bool_network/abstract_models/yeast.h File Reference

Definition of an interface representing a yeast's model.
`#include <ostream>`

### Classes

- class bn::abstract_models::yeast
  *Yeast model.*

### 2.4.1 Detailed Description

Definition of an interface representing a yeast's model.
Definition in file yeast.h.

## 2.5 include/bool_network/dynamic/abstract_model.h File Reference

Definition of a class representing an abstract model of boolean network.

```
#include <cstddef>
#include <bitset>
#include <ctime>
```

### Classes

- class bn::dynamic::abstract_model< Size >
  *Abstract model of a boolean network.*

### 2.5.1 Detailed Description

Definition of a class representing an abstract model of boolean network.
Definition in file abstract_model.h.

## 2.6 include/bool_network/dynamic/matrix_model.h File Reference

Definition of a class representing a model of boolean network with the rule contained in a matrix.

```
#include <cstddef>
#include <cstdlib>
#include "bool_network/dynamic/abstract_model.h"
```

### Classes

- class bn::dynamic::matrix_model< Size, Coef >
  *Model of a boolean network based on a matrix of transition.*

### 2.6.1 Detailed Description

Definition of a class representing a model of boolean network with the rule contained in a matrix.
Definition in file matrix_model.h.

## 2.7 include/bool_network/dynamic/state_machine.h File Reference

Definition of a class representing a state machine.

```
#include <vector>
#include <iterator>
#include <algorithm> #include <cstddef>
#include <ctime>
```

### Classes

- class bn::dynamic::state_machine< Model >
  *State machine.*

### 2.7.1 Detailed Description

Definition of a class representing a state machine.
Definition in file state_machine.h.

## 2.8 include/bool_network/dynamic/timed_matrix_model.h File Reference

Definition of a class representing a model of Boolean network with the rule (using time) contained in a matrix.

```
#include "bool_network/dynamic/matrix_model.h"
```

**Classes**

- struct bn::dynamic::timed_coef
  *Matrix's coefficient with a time retard on the effect.*
- class bn::dynamic::matrix_model< Size, timed_coef >
- struct bn::dynamic::timed_matrix_model< Size >
  *Timed model of a boolean network.*

### 2.8.1 Detailed Description

Definition of a class representing a model of Boolean network with the rule (using time) contained in a matrix.
Definition of a specialization of the base class matrix_model. This one let use a matrix to perform a model transformation by using the time as a retard on the original effect attempted.
Definition in file timed_matrix_model.h.

## 2.9 `include/bool_network/simulation/basic.h` File Reference

Definition of a class to simply simulate a state machine.

```
#include <cstddef>
```

**Classes**

- class bn::simulation::basic< Machine, Model, Unit >
  *Basic simulation of the state machine.*

### 2.9.1 Detailed Description

Definition of a class to simply stimulate a state machine.
Definition in file basic.h.

## 2.10 `include/bool_network/simulation/converge.h` File Reference

Definition of a class to simulate a state machine with the trace of the passage.

```
#include <map>
#include <cstddef>
```

**Classes**

- class bn::simulation::converge< Machine, Model, Unit >
  *Simulation of a state machine that keeps a trace of the passage.*

### 2.10.1 Detailed Description

Definition of a class to simulate a state machine with the trace of the passage.
Definition in file converge.h.