LOICA

Release 1.0.6

Gonzalo Vidal

Mar 23, 2022

CONTENTS

CHAPTER

ONE

INTRODUCTION

Welcome to the LOICA (Logical Operators for Integrated Cell Algorithms) repository, our Python package for designing, modeling and characterizing genetic networks.

As you may have noticed, our logo features a beautiful bird—loica (Leistes loyca); a bird native to Chile known for its particular red chest and legendary kindness, with which we share name.

CHAPTER TWO

INSTALLATION

Installing LOICA is way easier than pronuncing it!

```
pip install loica
```

For more details please refer to our Wiki for installation instructions and developer guides.

CHAPTER

THREE

LOICA ALLOWS YOU TO:

- Compile Code into DNA fragments that execute Cell Algorithms
- Easy programation of genetic network models
- Generation of synthetic data
- Communicate with Flapjack
- Use and output SBOL files
- Use all sorts of cellular computation
- Easy, fluid and customisable DNA design

CHAPTER

FOUR

TUTORIALS

Now that you have LOICA installed you can familiarize yourself with the tool using the Jupyter notebook tutorials designed for this purpose.

CHAPTER

FIVE

API REFERENCE

This page contains auto-generated API reference documentation[1].

5.1 loica

5.1.1 Subpackages

loica.operators

Submodules

loica.operators.hill1

Module Contents

Classes

Hill1	A class that represents a DNA fragment that encode a genetic operator.

class Hill1(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: *loica.operators.operator.Operator*

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

. . .

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

[1] Created with sphinx-autoapi

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)
Return str(self).

`characterize`(*self*, *flapjack*, *receiver*, *inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *a_j*, *b_j*, *n_i=2*, *K_i=1*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *odval=[1] * 100*, *gamma=0*, *p0_1=0*, *p0_2=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *gamma*)

`loica.operators.hill2`

Module Contents

Classes

Hill2	A class that represents a DNA fragment that encode a genetic operator.

class Hill2(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='orange'*)
Bases: *loica.operators.operator.Operator*

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

. . .

input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

output [Regulator | Reporter | List] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)
Return str(self).

`characterize`(*self*, *flapjack*, *receiver1*, *receiver2*, *chemical1*, *chemical2*, *nor_inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*, *lower_bounds=[0] * 8*, *upper_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0]*, *init_x=[1, 2, 1, 2, 1, 0, 0, 0]*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *rep1_K=1*, *rep1_n=2*, *rep2_K=1*, *rep2_n=2*, *alpha0=1*, *alpha1=0*, *alpha2=0*, *alpha3=0*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *a_B=100.0*, *b_B=0*, *K_B=1*, *n_B=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *B=0*, *odval=[1] * 100*, *gamma=0*, *rep1_0=0*, *rep2_0=0*, *fp_0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *a_B*, *b_B*, *K_B*, *n_B*, *chem1*, *chem2*, *gamma*)

`loica.operators.operator`

Module Contents

Classes

`Operator`	A class that represents a DNA fragment that encode a genetic operator.

class `Operator`(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

`__str__`(*self*)
Return str(self).

`loica.operators.receiver`

Module Contents

Classes

`Receiver`	A class that represents a DNA fragment that encode a genetic operator.

`class Receiver(`*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*`)`

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

…

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__(`*self*`)`

`characterize(`*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*`)`

`expression_rate(`*self*, *t*, *dt*`)`

`forward_model(`*self*, *a=0*, *b=1*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=[0]*, *odval=[1] * 100*, *gamma=0*, *p0=0*, *nt=100*`)`

`residuals(`*self*, *df*, *oddf*`)`

`loica.operators.source`

Module Contents

Classes

Source	A class that represents a DNA fragment that encode a genetic operator.

`class Source(`*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*`)`

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

…

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*)

Package Contents

Classes

Hill1	A class that represents a DNA fragment that encode a genetic operator.
Hill2	A class that represents a DNA fragment that encode a genetic operator.
Operator	A class that represents a DNA fragment that encode a genetic operator.
Operator	A class that represents a DNA fragment that encode a genetic operator.
Operator	A class that represents a DNA fragment that encode a genetic operator.
Operator	A class that represents a DNA fragment that encode a genetic operator.
Operator	A class that represents a DNA fragment that encode a genetic operator.
Receiver	A class that represents a DNA fragment that encode a genetic operator.
Receiver	A class that represents a DNA fragment that encode a genetic operator.
Receiver	A class that represents a DNA fragment that encode a genetic operator.
Source	A class that represents a DNA fragment that encode a genetic operator.
Source	A class that represents a DNA fragment that encode a genetic operator.
Source	A class that represents a DNA fragment that encode a genetic operator.

continues on next page

Table 6 – continued from previous page

Source	A class that represents a DNA fragment that encode a genetic operator.

class Hill1(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: *loica.operators.operator.Operator*

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

. . .

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

Return str(self).

characterize(*self*, *flapjack*, *receiver*, *inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *a_j*, *b_j*, *n_i=2*, *K_i=1*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *odval=[1] * 100*, *gamma=0*, *p0_1=0*, *p0_2=0*, *nt=100*)

residuals(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *gamma*)

class Hill2(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='orange'*)

Bases: *loica.operators.operator.Operator*

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

. . .

input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

output [Regulator | Reporter | List] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)
Return str(self).

`characterize`(*self*, *flapjack*, *receiver1*, *receiver2*, *chemical1*, *chemical2*, *nor_inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*, *lower_bounds=[0] * 8*, *upper_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0]*, *init_x=[1, 2, 1, 2, 1, 0, 0, 0]*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *rep1_K=1*, *rep1_n=2*, *rep2_K=1*, *rep2_n=2*, *alpha0=1*, *alpha1=0*, *alpha2=0*, *alpha3=0*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *a_B=100.0*, *b_B=0*, *K_B=1*, *n_B=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *B=0*, *odval=[1] * 100*, *gamma=0*, *rep1_0=0*, *rep2_0=0*, *fp_0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *a_B*, *b_B*, *K_B*, *n_B*, *chem1*, *chem2*, *gamma*)

`class Operator`(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
A class that represents a DNA fragment that encode a genetic operator.

…

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

`__str__`(*self*)
Return str(self).

`class Operator`(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
A class that represents a DNA fragment that encode a genetic operator.

…

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

`__str__`(*self*)
Return str(self).

`class Operator`(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
A class that represents a DNA fragment that encode a genetic operator.

…

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

`__str__`(*self*)
Return str(self).

`class Operator`(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
A class that represents a DNA fragment that encode a genetic operator.

. . .

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

`__str__`(*self*)
Return str(self).

`class Operator`(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
A class that represents a DNA fragment that encode a genetic operator.

. . .

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

`__str__`(*self*)
Return str(self).

`class Receiver`(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

. . .

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *a=0*, *b=1*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=[0]*, *odval=[1] * 100*, *gamma=0*, *p0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*)

`class Receiver`(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *a=0*, *b=1*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=[0]*, *odval=[1] * 100*, *gamma=0*, *p0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*)

`class Receiver`(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *a=0*, *b=1*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=[0]*, *odval=[1] * 100*, *gamma=0*, *p0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*)

`class Source`(*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*)

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

forward_model(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*)

residuals(*self*, *df*, *oddf*)

class **Source**(*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*)

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

characterize(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*)

residuals(*self*, *df*, *oddf*)

class **Source**(*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*)

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

characterize(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*)

residuals(*self*, *df*, *oddf*)

class Source(*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*)

Bases: *Operator*

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

characterize(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*)

residuals(*self*, *df*, *oddf*)

5.1.2 Submodules

loica.assay

Module Contents

Classes

Assay	Assay measures a set of samples in parallel at a set of timepoints.

class Assay(*samples*, *n_measurements*, *interval*, *name='Loica assay'*, *description=''*, *biomass_signal_id=None*)

Assay measures a set of samples in parallel at a set of timepoints. Connects to flapjack to generate data, and to fit parameters to data.

...

samples [List[Sample]] List of Samples that belongs to the Assay

n_measurements [int] Number of measurements to take

interval [int] Time in hours between each measurements

name [str] Name of the Assay

description: str Descriptioin of the Assay

biomass_signal_id [int] Flapjack ID of the Assay that is associated with the Assay

run(substeps=10, nsr=0, biomass_bg=0, fluo_bg=0) Runs the Assay time series

upload(flapjack, study) Upload the data produced by running the Assay to Flapjack into the Study

Assay measures a set of samples in parallel at a set of timepoints Connects to flapjack to generate data, and to fit parameters to data

run(*self*, *substeps=10*, *nsr=0*, *biomass_bg=0*, *fluo_bg=0*, *stochastic=False*)
Run the assay measuring at specified time points, with simulation time step dt

upload(*self*, *flapjack*, *study*)

loica.colony

Module Contents

Classes

Colony	

class **Colony**(*circuit=None*, *r0=1*, *mu0=1*)

fun(*self*, *x*)

kymograph(*self*, *nx*, *t0*, *tmax*)

map_kymo(*self*, *kymo*)

norm_kymo(*self*, *kymo*)

loica.geneproduct

Module Contents

Classes

GeneProduct	A class that represents a gene product, protein or RNA.
Regulator	Representation of a regulatory gene product.
Reporter	Representation of a regulatory gene product.

class **GeneProduct**(*name*, *init_concentration=0*, *degradation_rate=0*, *uri=None*, *sbol_comp=None*, *type_='PRO'*, *color='silver'*)

A class that represents a gene product, protein or RNA.

...

name [str] Name of the gene product

init_concentration [int | float] Initial concentration of the gene product in Molar

degradation_rate [int | float] Degradation rate of the gene product

type_ [str, optional] Molecular type of the gene product, could be 'PRO' or 'RNA'

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

`shape = ^`

`__str__`(*self*)
Return str(self).

`express`(*self*, *rate*)

`initialize`(*self*)

`step`(*self*, *growth_rate*, *dt*)

`class Regulator`(*name*, *init_concentration=0*, *degradation_rate=0*, *sbol_comp=None*, *color='lightgreen'*)
Bases: `GeneProduct`

Representation of a regulatory gene product. Child of GeneProduct.

`class Reporter`(*name*, *init_concentration=0*, *degradation_rate=0*, *signal_id=None*, *color='w'*, *sbol_comp=None*)
Bases: `GeneProduct`

Representation of a regulatory gene product.

signal_id [str, optional] Flapjack ID of the signal that the reporter is associated with.

color [str, optional] Color of the reporter

`loica.genetic_network`

Module Contents

Classes

`GeneticNetwork`	Representation of a genetic netowrk composed by a set of Operators, Regulators and Reporters.

`class GeneticNetwork`(*vector=None*)
Representation of a genetic netowrk composed by a set of Operators, Regulators and Reporters.

...

operators [List[Operator]] List of Operators that are part of the genetic network

regulators [List[Regulator]] List of Regulators that are part of the genetic network

reporters [List[Reporter]] List of Reporters that are part of the genetic network

vector [int] Flapjack ID of the vector that is associated with the genetic network

to_graph() Builds a graph representation of the genetic netwok

draw() Generates a plot of the graph representation builded by to_graph()

to_sbol(sbol_doc=None) Generates a SBOL3 Document representation of the genetic network on sbol_doc

`add_operator`(*self*, *ops*)

add_regulator(*self*, *regs*)

add_reporter(*self*, *reps*)

draw(*self*, *node_shape='o'*, *node_size=500*, *linewidths=0*, *alpha=0.5*, *arrowsize=10*, *font_size=6*, *font_family='Tahoma'*, *font_weight='bold'*, *pos=nx.kamada_kawai_layout*, *contracted=False*)

initialize(*self*)

step(*self*, *growth_rate=1*, *t=0*, *dt=0.1*)

step_stochastic(*self*, *growth_rate=1*, *t=0*, *dt=0.1*)

substep_stochastic(*self*, *t=0*, *dt=0.1*, *growth_rate=1*)

to_contracted_graph(*self*)

to_graph(*self*)

to_sbol(*self*, *sbol_doc: sbol3.Document = None*) → sbol3.Document

Convert the genetic network to SBOL. :param sbol_doc: The SBOL document to add the genetic network to.

`loica.metabolism`

Module Contents

Classes

DataMetabolism	Characterized context for gene expression, incorporates biomass and growth rate.
Metabolism	Context for gene expression, incorporates biomass and growth rate.
SimulatedMetabolism	Simulated context for gene expression, incorporates biomass and growth rate.

Functions

gompertz(t, y0, ymax, um, l)	
gompertz_growth_rate(t, y0, ymax, um, l)	
ramp_biomass(t, od0, start, slope)	
ramp_growth_rate(t, start, slope)	
step_biomass(t, od0, start)	
step_growth_rate(t, start)	

class DataMetabolism(*name*, *fj*, *media*, *strain*, *vector*, *biomass_signal*)

Bases: *Metabolism*

Characterized context for gene expression, incorporates biomass and growth rate. …

name [str, optional] Name of the metabolism or correponding strain

fj [Flapjack] Flapjack instance used to fetch data from

media [str] Name of the media to query

strain [str] Name of the strain to query

vector [str] Name of the vector to query

biomass_signal [str] Name of signal to query and use as biomass

biomass(t) Return biomass at a given time from characterization data

growth:rate(t) Return growth rate at a given time from characterization data

`biomass`(*self*, *t*)

`growth_rate`(*self*, *t*)

`class Metabolism`(*name=None*)

Context for gene expression, incorporates biomass and growth rate. …

name [str, optional] Name of the metabolism or correponding strain

`class SimulatedMetabolism`(*name*, *biomass*, *growth_rate*)

Bases: `Metabolism`

Simulated context for gene expression, incorporates biomass and growth rate. …

name [str, optional] Name of the metabolism or correponding strain

biomass A function of time that describes biomass f(t)=biomass

growth_rate A function of time that describes the growth rate f(t)=growth rate

`gompertz`(*t*, *y0*, *ymax*, *um*, *l*)

`gompertz_growth_rate`(*t*, *y0*, *ymax*, *um*, *l*)

`ramp_biomass`(*t*, *od0*, *start*, *slope*)

`ramp_growth_rate`(*t*, *start*, *slope*)

`step_biomass`(*t*, *od0*, *start*)

`step_growth_rate`(*t*, *start*)

`loica.sample`

Module Contents

Classes

`Sample`	Representation of a sample that encapsulates GeneticNetwork and Metabolism.

`class Sample`(*genetic_network=None*, *metabolism=None*, *assay=None*, *media=None*, *strain=None*)

Representation of a sample that encapsulates GeneticNetwork and Metabolism. Incorporate environment information such as Supplements or chemicals, strain and media. Ex: 1 well in a plate, single cell. …

genetic_network [GeneticNetwork] genetic network that is part of the sample

metabolism [Metabolism] metabolism that drives the genetic network in the sample

assay [Assay] assay to which this sample belongs

media [str] Name of the media in the sample

strain [str]

Name of the strain in the sample

Methods

add_supplement(supplement, concentration) stablishes the concentration of Supplement

initialize(*self*)

set_regulator(*self*, *name*, *concentration*)

set_reporter(*self*, *name*, *concentration*)

set_supplement(*self*, *supplement*, *concentration*)

step(*self*, *t*, *dt*, *stochastic=False*)

loica.supplement

Module Contents

Classes

Supplement	Representation of a chemical

class Supplement(*name*, *pubchemid=None*, *supplier_id=None*, *sbol_comp=None*, *color='pink'*)

Representation of a chemical

...

name [str] Name of the supplement

concentration [int | float] concentration of the supplement in Molar

pubchemid [str] PubChemID URI of the supplement

supplier_id [str] Supplier ID of the supplement. An URL of the product that you aquire. Accepts list of the form [product URL, catalog number, batch].

sbol_comp [str] SBOL component of the supplement.

__str__(*self*)

Return str(self).

loica.util

Module Contents

Functions

characterize_growth(flapjack, vector, media, strain, biomass_signal, n_gaussians, epsilon)	
forward_model_growth(Dt=0.05, sim_steps=10, muval=[0] * 100, od0=0, nt=100)	
load_loica(filename)	
residuals_growth(data, epsilon, dt, t, n_gaussians)	
save_loica(obj, filename)	

characterize_growth(*flapjack*, *vector*, *media*, *strain*, *biomass_signal*, *n_gaussians*, *epsilon*)

forward_model_growth(*Dt=0.05*, *sim_steps=10*, *muval=[0] * 100*, *od0=0*, *nt=100*)

load_loica(*filename*)

residuals_growth(*data*, *epsilon*, *dt*, *t*, *n_gaussians*)

save_loica(*obj*, *filename*)

5.1.3 Package Contents

Classes

Assay	Assay measures a set of samples in parallel at a set of timepoints.
Colony	
DataMetabolism	Characterized context for gene expression, incorporates biomass and growth rate.
GeneProduct	A class that represents a gene product, protein or RNA.
GeneticNetwork	Representation of a genetic netowrk composed by a set of Operators, Regulators and Reporters.
Hill1	A class that represents a DNA fragment that encode a genetic operator.
Hill1	A class that represents a DNA fragment that encode a genetic operator.
Hill2	A class that represents a DNA fragment that encode a genetic operator.
Hill2	A class that represents a DNA fragment that encode a genetic operator.
Metabolism	Context for gene expression, incorporates biomass and growth rate.

continues on next page

Table 16 – continued from previous page

Operator	A class that represents a DNA fragment that encode a genetic operator.
Operator	A class that represents a DNA fragment that encode a genetic operator.
Receiver	A class that represents a DNA fragment that encode a genetic operator.
Receiver	A class that represents a DNA fragment that encode a genetic operator.
Regulator	Representation of a regulatory gene product.
Regulator	Representation of a regulatory gene product.
Reporter	Representation of a regulatory gene product.
Reporter	Representation of a regulatory gene product.
Sample	Representation of a sample that encapsulates GeneticNetwork and Metabolism.
SimulatedMetabolism	Simulated context for gene expression, incorporates biomass and growth rate.
Source	A class that represents a DNA fragment that encode a genetic operator.
Source	A class that represents a DNA fragment that encode a genetic operator.
Supplement	Representation of a chemical
Supplement	Representation of a chemical

Functions

characterize_growth(flapjack, vector, media, strain, biomass_signal, n_gaussians, epsilon)	
forward_model_growth(Dt=0.05, sim_steps=10, muval=[0] * 100, od0=0, nt=100)	
gompertz(t, y0, ymax, um, l)	
gompertz_growth_rate(t, y0, ymax, um, l)	
load_loica(filename)	
ramp_biomass(t, od0, start, slope)	
ramp_growth_rate(t, start, slope)	
residuals_growth(data, epsilon, dt, t, n_gaussians)	
save_loica(obj, filename)	
step_biomass(t, od0, start)	
step_growth_rate(t, start)	

class Assay(*samples*, *n_measurements*, *interval*, *name='Loica assay'*, *description=''*, *biomass_signal_id=None*)

Assay measures a set of samples in parallel at a set of timepoints. Connects to flapjack to generate data, and to fit parameters to data.

…

samples [List[Sample]] List of Samples that belongs to the Assay

n_measurements [int] Number of measurements to take

interval [int] Time in hours between each measurements

name [str] Name of the Assay

description: str Descriptioin of the Assay

biomass_signal_id [int] Flapjack ID of the Assay that is associated with the Assay

run(substeps=10, nsr=0, biomass_bg=0, fluo_bg=0) Runs the Assay time series

upload(flapjack, study) Upload the data produced by running the Assay to Flapjack into the Study

Assay measures a set of samples in parallel at a set of timepoints Connects to flapjack to generate data, and to fit parameters to data

`run`(*self*, *substeps=10*, *nsr=0*, *biomass_bg=0*, *fluo_bg=0*, *stochastic=False*)
Run the assay measuring at specified time points, with simulation time step dt

`upload`(*self*, *flapjack*, *study*)

`class Colony`(*circuit=None*, *r0=1*, *mu0=1*)

`fun`(*self*, *x*)

`kymograph`(*self*, *nx*, *t0*, *tmax*)

`map_kymo`(*self*, *kymo*)

`norm_kymo`(*self*, *kymo*)

`class DataMetabolism`(*name*, *fj*, *media*, *strain*, *vector*, *biomass_signal*)
Bases: `Metabolism`

Characterized context for gene expression, incorporates biomass and growth rate. …

name [str, optional] Name of the metabolism or correponding strain

fj [Flapjack] Flapjack instance used to fetch data from

media [str] Name of the media to query

strain [str] Name of the strain to query

vector [str] Name of the vector to query

biomass_signal [str] Name of signal to query and use as biomass

biomass(t) Return biomass at a given time from characterization data

growth:rate(t) Return growth rate at a given time from characterization data

`biomass`(*self*, *t*)

`growth_rate`(*self*, *t*)

`class GeneProduct`(*name*, *init_concentration=0*, *degradation_rate=0*, *uri=None*, *sbol_comp=None*, *type_='PRO'*, *color='silver'*)
A class that represents a gene product, protein or RNA.

…

name [str] Name of the gene product

init_concentration [int | float] Initial concentration of the gene product in Molar

degradation_rate [int | float] Degradation rate of the gene product

type_ [str, optional] Molecular type of the gene product, could be 'PRO' or 'RNA'

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

`shape = ^`

`__str__`(*self*)
Return str(self).

`express`(*self*, *rate*)

`initialize`(*self*)

`step`(*self*, *growth_rate*, *dt*)

`class GeneticNetwork`(*vector=None*)
Representation of a genetic netowrk composed by a set of Operators, Regulators and Reporters.

...

operators [List[Operator]] List of Operators that are part of the genetic network

regulators [List[Regulator]] List of Regulators that are part of the genetic network

reporters [List[Reporter]] List of Reporters that are part of the genetic network

vector [int] Flapjack ID of the vector that is associated with the genetic network

to_graph() Builds a graph representation of the genetic netwok

draw() Generates a plot of the graph representation builded by to_graph()

to_sbol(sbol_doc=None) Generates a SBOL3 Document representation of the genetic network on sbol_doc

`add_operator`(*self*, *ops*)

`add_regulator`(*self*, *regs*)

`add_reporter`(*self*, *reps*)

`draw`(*self*, *node_shape='o'*, *node_size=500*, *linewidths=0*, *alpha=0.5*, *arrowsize=10*, *font_size=6*, *font_family='Tahoma'*, *font_weight='bold'*, *pos=nx.kamada_kawai_layout*, *contracted=False*)

`initialize`(*self*)

`step`(*self*, *growth_rate=1*, *t=0*, *dt=0.1*)

`step_stochastic`(*self*, *growth_rate=1*, *t=0*, *dt=0.1*)

`substep_stochastic`(*self*, *t=0*, *dt=0.1*, *growth_rate=1*)

`to_contracted_graph`(*self*)

`to_graph`(*self*)

`to_sbol`(*self*, *sbol_doc: sbol3.Document = None*) → sbol3.Document
Convert the genetic network to SBOL. :param sbol_doc: The SBOL document to add the genetic network to.

class Hill1(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: `loica.operators.operator.Operator`

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

Return str(self).

characterize(*self*, *flapjack*, *receiver*, *inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *a_j*, *b_j*, *n_i=2*, *K_i=1*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *odval=[1] * 100*, *gamma=0*, *p0_1=0*, *p0_2=0*, *nt=100*)

residuals(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *gamma*)

class Hill1(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: `loica.operators.operator.Operator`

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)
Return str(self).

`characterize`(*self*, *flapjack*, *receiver*, *inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *a_j*, *b_j*, *n_i=2*, *K_i=1*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *odval=[1] * 100*, *gamma=0*, *p0_1=0*, *p0_2=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *gamma*)

`class Hill2`(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='orange'*)
Bases: `loica.operators.operator.Operator`

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

…

input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

output [Regulator | Reporter | List] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)
Return str(self).

`characterize`(*self*, *flapjack*, *receiver1*, *receiver2*, *chemical1*, *chemical2*, *nor_inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*, *lower_bounds=[0] * 8*, *upper_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0]*, *init_x=[1, 2, 1, 2, 1, 0, 0, 0]*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *rep1_K=1*, *rep1_n=2*, *rep2_K=1*, *rep2_n=2*, *alpha0=1*, *alpha1=0*, *alpha2=0*, *alpha3=0*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *a_B=100.0*, *b_B=0*, *K_B=1*, *n_B=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *B=0*, *odval=[1] * 100*, *gamma=0*, *rep1_0=0*, *rep2_0=0*, *fp_0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *a_B*, *b_B*, *K_B*, *n_B*, *chem1*, *chem2*, *gamma*)

`class Hill2`(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='orange'*)
Bases: `loica.operators.operator.Operator`

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

. . .

input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

output [Regulator | Reporter | List] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)
Return str(self).

`characterize`(*self*, *flapjack*, *receiver1*, *receiver2*, *chemical1*, *chemical2*, *nor_inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*, *lower_bounds=[0] * 8*, *upper_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0]*, *init_x=[1, 2, 1, 2, 1, 0, 0, 0]*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *rep1_K=1*, *rep1_n=2*, *rep2_K=1*, *rep2_n=2*, *alpha0=1*, *alpha1=0*, *alpha2=0*, *alpha3=0*, *a_A=100.0*, *b_A=0*, *K_A=1*, *n_A=2*, *a_B=100.0*, *b_B=0*, *K_B=1*, *n_B=2*, *Dt=0.05*, *sim_steps=10*, *A=0*, *B=0*, *odval=[1] * 100*, *gamma=0*, *rep1_0=0*, *rep2_0=0*, *fp_0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*, *a_A*, *b_A*, *K_A*, *n_A*, *a_B*, *b_B*, *K_B*, *n_B*, *chem1*, *chem2*, *gamma*)

`class Metabolism`(*name=None*)
Context for gene expression, incorporates biomass and growth rate. . . .

name [str, optional] Name of the metabolism or correponding strain

`class Operator`(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)
A class that represents a DNA fragment that encode a genetic operator.

. . .

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

`__str__`(*self*)
Return str(self).

class Operator(*output*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

A class that represents a DNA fragment that encode a genetic operator.

. . .

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

__str__(*self*)

Return str(self).

class Receiver(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: *Operator*

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

. . .

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

characterize(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *a=0*, *b=1*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=[0]*, *odval=[1] * 100*, *gamma=0*, *p0=0*, *nt=100*)

residuals(*self*, *df*, *oddf*)

class Receiver(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: *Operator*

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

. . .

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *a=0*, *b=1*, *K_A=1*, *n_A=2*, *Dt=0.05*, *sim_steps=10*, *A=[0]*, *odval=[1] * 100*, *gamma=0*, *p0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*)

`class Regulator`(*name*, *init_concentration=0*, *degradation_rate=0*, *sbol_comp=None*, *color='lightgreen'*)
Bases: `GeneProduct`

Representation of a regulatory gene product. Child of GeneProduct.

`class Regulator`(*name*, *init_concentration=0*, *degradation_rate=0*, *sbol_comp=None*, *color='lightgreen'*)
Bases: `GeneProduct`

Representation of a regulatory gene product. Child of GeneProduct.

`class Reporter`(*name*, *init_concentration=0*, *degradation_rate=0*, *signal_id=None*, *color='w'*, *sbol_comp=None*)
Bases: `GeneProduct`

Representation of a regulatory gene product.

signal_id [str, optional] Flapjack ID of the signal that the reporter is associated with.

color [str, optional] Color of the reporter

`class Reporter`(*name*, *init_concentration=0*, *degradation_rate=0*, *signal_id=None*, *color='w'*, *sbol_comp=None*)
Bases: `GeneProduct`

Representation of a regulatory gene product.

signal_id [str, optional] Flapjack ID of the signal that the reporter is associated with.

color [str, optional] Color of the reporter

`class Sample`(*genetic_network=None*, *metabolism=None*, *assay=None*, *media=None*, *strain=None*)
Representation of a sample that encapsulates GeneticNetwork and Metabolism. Incorporate environment information such as Supplements or chemicals, strain and media. Ex: 1 well in a plate, single cell. . . .

genetic_network [GeneticNetwork] genetic network that is part of the sample

metabolism [Metabolism] metabolism that drives the genetic network in the sample

assay [Assay] assay to which this sample belongs

media [str] Name of the media in the sample

strain [str]

Name of the strain in the sample

Methods

add_supplement(supplement, concentration) stablishes the concentration of Supplement

`initialize`(*self*)

`set_regulator`(*self*, *name*, *concentration*)

`set_reporter`(*self*, *name*, *concentration*)

`set_supplement`(*self*, *supplement*, *concentration*)

`step`(*self*, *t*, *dt*, *stochastic=False*)

`class` **`SimulatedMetabolism`**(*name*, *biomass*, *growth_rate*)

Bases: `Metabolism`

Simulated context for gene expression, incorporates biomass and growth rate. …

name [str, optional] Name of the metabolism or correponding strain

biomass A function of time that describes biomass f(t)=biomass

growth_rate A function of time that describes the growth rate f(t)=growth rate

`class` **`Source`**(*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*)

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

…

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*) `residuals`(*self*, *df*, *oddf*)

`class Source`(*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*)

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

`__str__`(*self*)

`characterize`(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

`expression_rate`(*self*, *t*, *dt*)

`forward_model`(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*)

`residuals`(*self*, *df*, *oddf*)

`class Supplement`(*name*, *pubchemid=None*, *supplier_id=None*, *sbol_comp=None*, *color='pink'*)

Representation of a chemical

...

name [str] Name of the supplement

concentration [int | float] concentration of the supplement in Molar

pubchemid [str] PubChemID URI of the supplement

supplier_id [str] Supplier ID of the supplement. An URL of the product that you aquire. Accepts list of the form [product URL, catalog number, batch].

sbol_comp [str] SBOL component of the supplement.

`__str__`(*self*)

Return str(self).

`class Supplement`(*name*, *pubchemid=None*, *supplier_id=None*, *sbol_comp=None*, *color='pink'*)

Representation of a chemical

...

name [str] Name of the supplement

concentration [int | float] concentration of the supplement in Molar

pubchemid [str] PubChemID URI of the supplement

supplier_id [str] Supplier ID of the supplement. An URL of the product that you aquire. Accepts list of the form [product URL, catalog number, batch].

sbol_comp [str] SBOL component of the supplement.

`__str__`(*self*)
Return str(self).

`characterize_growth`(*flapjack*, *vector*, *media*, *strain*, *biomass_signal*, *n_gaussians*, *epsilon*)

`forward_model_growth`(*Dt=0.05*, *sim_steps=10*, *muval=[0] * 100*, *od0=0*, *nt=100*)

`gompertz`(*t*, *y0*, *ymax*, *um*, *l*)

`gompertz_growth_rate`(*t*, *y0*, *ymax*, *um*, *l*)

`load_loica`(*filename*)

`ramp_biomass`(*t*, *od0*, *start*, *slope*)

`ramp_growth_rate`(*t*, *start*, *slope*)

`residuals_growth`(*data*, *epsilon*, *dt*, *t*, *n_gaussians*)

`save_loica`(*obj*, *filename*)

`step_biomass`(*t*, *od0*, *start*)

`step_growth_rate`(*t*, *start*)

CHAPTER

SIX

INTRODUCTION

Welcome to the LOICA (Logical Operators for Integrated Cell Algorithms) repository, our Python package for designing, modeling and characterizing genetic networks.

As you may have noticed, our logo features a beautiful bird—loica (Leistes loyca); a bird native to Chile known for its particular red chest and legendary kindness, with which we share name.

CHAPTER

SEVEN

INSTALLATION

Installing LOICA is way easier than pronuncing it!

```
pip install loica
```

For more details please refer to our Wiki for installation instructions and developer guides.

CHAPTER

EIGHT

LOICA ALLOWS YOU TO:

- Compile Code into DNA fragments that execute Cell Algorithms
- Easy programation of genetic network models
- Generation of synthetic data
- Communicate with Flapjack
- Use and output SBOL files
- Use all sorts of cellular computation
- Easy, fluid and customisable DNA design

CHAPTER

NINE

TUTORIALS

Now that you have LOICA installed you can familiarize yourself with the tool using the Jupyter notebook tutorials designed for this purpose.

PYTHON MODULE INDEX

INDEX