SQUAD

june 11, 2016

Type R package
Title
Version
Imports BoolNet, deSolve
Requirements
Description Automatically transform Boolean Regulatory Network Models to a continuous one using SQUAD formalism.
License
LazyLoad
Encoding
Author
Mainteiner
NeedsCompilation
Repository
Date/Publication

`asContinuous`	*converts a boolean regulatory model to a continuous model*

Description

Take a net object of class "BooleanNetwork" as that defined by loadNetwork() and return a continuous regulatory network model in the form of a net object of the class "SQUAD". The continuous network can be used as an object function argument for the deSolve R package.

Usage

asContinuous(net)

Example

```
> data(cellCycle)
# setting initial state
> state<-generateState(cellCycle,specs=c("CycA"=1,"CycD"=1))
# setting intervals for numerical integration
> times <- seq(min=0,max=1,seq.out=100)

> cellCycle.s <- asContinuous(cellCycle)
> simulations <- ode(y=state, fun=cellCycle.s$fun, times=times,
                    parameters = "default")
> # note that cellCycle$fun can be given as the function
> # parameter to ode solver method
> simulations
```

Details

The continuous network object retrieved by the function represent a system

of ordinary equations according to the methodology described in Martínez-Sosa, 2013 [1]. This methodology have the same fixed point attractors of that in the Boolean model but have the advantage that it takes into account kinetic parameters.

`graphToModel`	*transforms a connectivity Matrix to a dynamic regulatory network model*

Description

Takes a connectivity matrix that represent a static regulatory network graph and transforms it to a "BooleanNetwork" or "SQUAD" object that can be used to simulate network transitions and calculating attractors.

Usage

```
graphToModel(connectivityMatrix)
```

Arguments

connectivityMatrix	A square matrix of the nodes of the network with -1, 0, or 1 integer values that represent negative, null or positive regulation.

Example

```
# generate a sample random connectivity matrix
> net<-randomMatrix.e(dimensions=20)
> plotAttractors(getAttractors(net))
```

Details

Connectivity matrices are defined as that with entries A_{ij} representing

a regulation from the node i to j. Positive (negative) regulations are represent by 1 (-1). No regulatory dependencies are indicated by zero. The model assumes that node j activation is achieved if and only if there is at least one of their positive regulators is active and all negative regulators inputs are inactive.

`loadNetWorkSQUAD` *loads a regulatory network given as a SQUAD format*

Description

Load a regulatory network model given as a SQUAD format. It returns an object of the class "SQUAD" that can be used to make simulations using squad().

Usage

loadNetworkSQUAD("file.txt")

Arguments

file	A text file of the format "SQUAD". See details for the definition of the "SQUAD" format.
fixed	A list of of named numerical variables. Variables belongs to the [0,1] interval. A valid fixed parameter is list("$gen1" = 0, "gen2" = 0.5, ...$) where the length of the list is less that the number of nodes in the regulatory network model.

Example

```
# loading a "SQUAD" model from the SQUAD format
> net <- loadNetworkSQUAD("file.txt")
> simulations <- ode(y=state, fun=net$fun, times=times,
                     parameters = "default")
> # note that cellCycle$fun can be given as the function
```

```
> # parameter to ode solver method
> simulations
```

Details

An example of the squad format has the following structure:

targets; factors
A; A
B; $min(A, 1 - C)$
C; $1 - max(A, B)$

Where the first line defines is a header which indicates that the file is in squad format. The next lines defines the names of the nodes followed by a ";". The rest of the line is a diffuse logic expression which defines the ω parameter of the squad function, see [1]. It returns a object of the class "SQUAD" that can be used to perform simulations using the squad() function.

`getAttractorsAsynchronous` *calculates all asynchronous stationary states of a Boolean Regulatory Network*

Description

Finds all asynchronous stationary states of a Boolean Regulatory Network model. Takes as arguments an object of the "BoolNet" (or "SQUAD" under construction) and returns a data frame which contains the attractors of the model.

Usage

getAttractorsAsynchronous(net)

Arguments

net	An object of the "BoolNet" (or "SQUAD") class.

Example

```
> data(cellCycle)
> attractors <- getAttractorsAsynchronous(cellCycle)
> attractors
```

Details

The function is straightforward. It calls the ginsim library written by Nadi and colleagues, 2009 [2].

`squad`	*performs simulations of regulatory network models using the SQUAD methodology*

Description

Performs a variety simulations of Regulatory network models using the SQUAD methodology. It can be used to set up different initial states, parameters and also to simulate gain/loss of function mutants. It returns

Usage

squad(net,initialState,parameters,fixedGenes)

Arguments

net	An object of the "BoolNet" or "SQUAD" class.

Example

```
> data(cellCycle)
> net.sq <- asContinuous(cellCycle)
> initialState <- runif(length(cellCycle$genes),0,1)
> times <- seq(0,10,by=0.5)
> h <- rep(50,length(cellCycle$genes))
> gamma <- rep(1,length(cellCycle$genes))
> parameters <- list(h,gamma)
> squad(net.sq)
```

Details

It performs simulations according to the updated version of the Standardized Qualitative Analysis of Dynamical Systems (SQUAD) method described in Martínez-Sosa in 2013 [1], as modified from the first version in Di Cara, 2007 [2].

Bibliography

[1] P. Martínez-Sosa and L. Mendoza. The regulatory network that controls the differentiation of t lymphocytes. *Biosystems*, 113(2):96–103, 2013.

[2] A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya. Logical modelling of regulatory networks with ginsim 2.3. *Biosystems*, 97(2):134–139, 2009.