Introduction

MaBoSS is a C++ software for simulating continuous/discrete time Markov processes, applied on a Boolean network. MaBoSS uses a specific grammar for associating transition rates to each node. Given some initial conditions, MaBoSS applies Monte-Carlo kinetic algorithm (or Gillespie algorithm) to the network to produce time trajectories. Time evolution of probabilities are estimated. In addition, global and semi-global characterizations of the whole system are computed.

Synopsis

Running a MaBoSS simulation:

`MaBoSS -c` *CONFIGURATION_FILE* `-o` *OUTPUT_PREFIX NETWORK_DESCRIPTION_FILE*

or, using long options:

`MaBoSS --config` *CONFIGURATION_FILE* `--output` *OUTPUT_PREFIX NETWORK_DESCRIPTION_FILE*

Arguments:

- *NETWORK_DESCRIPTION_FILE*: file containing the network description

- *CONFIGURATION_FILE*: (optional) configuration file to be used for simulation; -c option can be used multiple times, configuration files being read in the given order

- *OUTPUT_PREFIX*: output prefix of the generated files

More options are described below.

Network Description File

Goals: description of the network: node names, logic and transition rates
Usual extension: .bnd

General Syntax

Each node is described as follows:

`node` *NODE_NAME* `{` *node_var1 = expr1; node_var2 = expr2; ...*`}`

Comments: same as C++ and Java: // or /* */

Expressions

An expression is an arithmetic and logical combination of node names (denoting node states), external variables (defined in the configuration files), current node variable values, integer or double literals.

Lexical units

Lexical Unit	Syntax	Regular Expression	Examples
node name and node variable	sequence of letters, digits and _ beginning by a letter or an _	[a-aA-Z_][a-aA-Z0-9_]*	MyNode p53
node variable value	when a node variable is used in an expression, its name must be prefixed by an @	@[a-aA-Z_][a-aA-Z_0-9]*	@logic @rate_up

Lexical Unit	Syntax	Regular Expression	Examples
external variable	sequence of letters, digits and _ beginning by a $	$[a-zA-Z_0-9]+	$myvar
integer literal	same lexical form as C, C++ and Java		10 2334
double literal	same lexical form as C, C++ and Java including scientific notation for doubles		1.23 1.2e+12

Operators

Supported operators in descending priority (same precedence as C, C++ and Java):

Operator	Aliases	Description
()		parenthesis
!	NOT	logical NOT
+ -		unary plus unary minus
* /		multiplication division
+ -		addition substraction
< <= > >=		less than less than or equal greater than greater than or equal
== !=		equal to not equal to
&&	& AND	logical and
||	| OR	logical or
^	XOR	logical or
? :		ternary conditional

Node Variables

Any node variable name can be defined and used, but three of them have semantics linked to the algorithm:

Node Variable	Description
rate_up	expression defining the transition rate to flip the state from 0 to 1
rate_down	expression defining the transition rate to flip the state from 1 to 0

logic	if rate_up and/or rate_down are not specified:
	rate_up = @logic ? 1.0 : 0.0;
	rate_down = @logic ? 0.0 : 1.0;

Example

```
// Basic network example

node A {
  rate_up = 1.1;
  rate_down = $A_rate_down * 10.2;
  // $A_rate_down external variable must be defined in one of the configuration files
}

node B {
  tmp = NOT A OR C;
```

```
  // note that node variable tmp has been introduced for convenience to define
  // rate_up and rate_down without code duplication

  rate_up = @tmp;
  rate_down = NOT @tmp ? $B_var * 12. : 0.;
  // $B_var must be defined in one of the configuration files
}

node C {
  rate_up = $var2 < 10 ? NOT B : A OR B;
  rate_down = A AND B;
  // $var2 must be defined in one of the configuration files
}

node D {
  logic = A OR (NOT B XOR C);
}
```

Configuration File

Goals: definition of estimation parameters, network parameters, output parameters and run parameters
Usual extension: .cfg

General Syntax

Each parameter assignment is as follows:

parameter = *value*;

value may be a boolean (TRUE or FALSE), an integer or a double

Comments: same as C++ and Java: // or /* */

Parameter list

Parameter	Type	Description	Default
node.istate	boolean or integer	initial state of the *node;* if value is an integer: - a positive not null value means a TRUE state; - a negative integer means that the initial state is random.	boolean random value
[node_list].istate	expression under the form: proba_expr1 [istate_list1], proba_expr2 [istate_list2] ...	initial probability distribution of the *node list* states: group of nodes given by *node list* is in the initial states given by istate_list# with respective probability of proba_expr# / (proba_expr1 + proba_expr2 +...); note that this can be applied to a single node, for instance: [A].istate = 0.3 [0], 0.7 [1]; or: [A].istate = $p0 [0], $p1 [1]; where $p0 and $p1 are variables defined in the configuration file or at the command line.	

Parameter	Type	Description	Default
node.is_internal	boolean	if set to TRUE, *node* is an internal node	FALSE
node.refstate	boolean	reference state of the *node* for computing Hamming distance distribution	not used if not specified
$*varname*	double, integer or boolean	numerical external variable to be used in expressions within the network description file; a variable can be also used in expressions within the configuration file, in the probabilities for initial states for instance	error if unspecified and used in the network description file
timetick	double	time window	0.5
max_time	double	maximum time for a trajectory	1000
sample_count	integer	number of trajectories	10000
discrete_time	boolean	if set to FALSE, continuous time is used, otherwise discrete time is used (jump process)	FALSE
use_physrandgen	boolean	if set to TRUE, the physical random generator /dev/urandom is used; a pseudo random generator is used otherwise; note that this physical random generator has poor performance in multi-threads	TRUE
seed_pseudorandom	integer	seed number when a pseudo random generator is used	0
display_traj	boolean	if set to TRUE, trajectories are displayed	FALSE
statdist_traj_count	integer	number of trajectories for stationary distribution clustering	0
statdist_cluster_threshold	double	threshold for stationary distribution clustering	1.0
statdist_similarity_cache_max_size	integer	a cache is used if the cluster trajectory number is less than or equal to this parameter	20000
thread_count	integer	number of threads to be used	1

Example

```
// these external variables can be used in the network description file (.bnd file)
$A_rate_down = 0.1;
$B_var = 2;
$var2 = 23;

// note: A, B, C and D must be nodes defined in the network description file
A.istate = 0;

$p0 = 1;
$p1 = 2;
$p2 = 1;
$p3 = 4;
[B, C].istate = $p0 [0, 0], $p1 [0, 1], $p2 [1, 0], $p3 [1, 1];
[D].istate = $p0 [0], ($p4*2) [1];

B.is_internal = 1;
```

```
C.is_internal = 1;
C.refstate = 1;

sample_count = 1000;
max_time = 100;
time_tick = 0.5;
discrete_time = 0;
use_physrandgen = FALSE;
seed_pseudorandom = 100;
statdist_traj_count = 10000;
thread_count = 4;
```

Building a model within MaBoSS

Inputs

Type	Parameters	Set in file
Network description	node names, network logic, transition rates	*NETWORK_DESCRIPTION_FILE*
Network parameters	initial conditions, discrete or continuous time, network variables (external variables)	*CONFIGURATION_FILE*
Estimation parameters	trajectory number, cluster trajectory number, cluster threshold, random generator hints	*CONFIGURATION_FILE*
Output parameters	time tick, trajectory length, internal nodes, value of reference nodes, trajectory display	*CONFIGURATION_FILE*
Run parameters	thread count	*CONFIGURATION_FILE*

Outputs

The generated files are as follows:

File name	Type	Optional	Description
*OUTPUT_PREFIX*_run.txt	Text file	No	Start and end time, summary of the parameters used, network display
*OUTPUT_PREFIX*_traj.txt	Text file	Yes. Generated if display_traj configuration parameter is set to TRUE	All trajectories: a trajectory is a sequence of the network state at each time tick
*OUTPUT_PREFIX*_probtraj.csv	Tabular file	No	Time evolution of probabilities: each line contains time, transition entropy, error on transition entropy, entropy, Hamming distance distribution, probabilities (and errors) for all states
*OUTPUT_PREFIX*_fp.csv	Tabular file	No	Fixed points with their probabilities
*OUTPUT_PREFIX*_statdist.csv	Tabular file	Yes. Generated if statdist_traj_count is greater than 0	- stationary distribution estimation for each trajectory, - clustering of stationary distribution estimations, - clustered estimation of stationary distributions (with errors),

Other Options

Generating logical boolean expressions

To generate the logical expressions from a network description file, use the -l or --generate-logical-expressions option as follows:

`MaBoSS -c` *CONFIGURATION_FILE* `-l` *NETWORK_DESCRIPTION_FILE*

or:

`MaBoSS --config` *CONFIGURATION_FILE* `--generate-logical-expressions` *NETWORK_DESCRIPTION_FILE*

Using this option on the previous network example will generate the following expressions:

```
A : !A
C : (!C & !B) | (C & !(A & B))
B : !B & (!A | C)
D : A | ((!B | C) & !(!B & C))
```

Generating a template configuration

To generate a complete configuration from a network description file, use the -t or --generate-config-template option as follows:

`MaBoSS -t` *NETWORK_DESCRIPTION_FILE*

or:

`MaBoSS --generate-config-template` *NETWORK_DESCRIPTION_FILE*

The generated output can be redirect to a file, edited and then used as a configuration file for a simulation.

Using this option on the previous network example will generate the following template configuration:

```
//
// MaBoSS 1.3.6 configuration template generated at Tue Nov  4 13:13:32 2014
//

// global configuration variables
time_tick = 0.5;
max_time = 1000;
sample_count = 10000;
discrete_time = 0;
use_physrandgen = 1;
seed_pseudorandom = 0;
display_traj = 0;
statdist_traj_count = 0;
statdist_cluster_threshold = 1;
thread_count = 1;
statdist_similarity_cache_max_size = 20000;

// variables to be set in the configuration file or by using the --config-vars option
$A_rate_down = 0;
$B_var = 0;
$var2 = 0;

// set is_internal attribute value to 1 if node is an internal node
A.is_internal = 0;
C.is_internal = 0;
B.is_internal = 0;
D.is_internal = 0;

// if node is a reference node, set refstate attribute value to 0 or 1 according to its ↩
    reference state
```

```
// if node is not a reference node, skip its refstate declaration or set value to -1
A.refstate = -1;
C.refstate = -1;
B.refstate = -1;
D.refstate = -1;

// if NODE initial state is:
// - equals to 1: NODE.istate = 1;
// - equals to 0: NODE.istate = 0;
// - random: NODE.istate = -1; OR [NODE].istate = 0.5 [0], 0.5 [1]; OR skip NODE.istate  ↩
    declaration
// - weighted random: [NODE].istate = P0 [0], P1 [1]; where P0 and P1 are arithmetic  ↩
    expressions
[A].istate = 0.5 [0], 0.5 [1];
[C].istate = 0.5 [0], 0.5 [1];
[B].istate = 0.5 [0], 0.5 [1];
[D].istate = 0.5 [0], 0.5 [1];
```

Other configuration options

Inline configuration expressions and variable definitions can be given with the command line:

- the -e (aka --config-expr) option is used to give an inline configuration expression.

- the -v (aka --config-vars) option is used to give the values of variables.

`MaBoSS -c` *CONFIGURATION_FILE* `-e` *CONFIG_EXPR* `-v` *VAR=NUMERIC* ...

or:

`MaBoSS --config` *CONFIGURATION_FILE* `--config-expr` *CONFIG_EXPR* `--config-vars` *VAR=NUMERIC* ...

Each configuration option -c (--config), -e (--config-expr) or -v (--config-vars) can be used multiple times:

- multiple -c (--config) and/or -e (--config-expr) options are managed in the order given at the command line,
- option -v (--config-vars) VAR=VALUE always overrides any VAR assignment in a configuration file or expression.

For instance:

```
MaBoSS -c conf1.txt -e 'thread_count = 2' -v '$var1=1.2,$var2=3.4' -c conf2.txt
       -e 'discrete_time = 0; time_tick = 0.1' -v '$var4=10'
```

Dumping a configuration

To dump a complete configuration from a network description file and one or more configuration files, use the -d or --dump-config option as follows:

`MaBoSS -c` *CONFIGURATION_FILE* `-d` *NETWORK_DESCRIPTION_FILE*

or:

`MaBoSS --config` *CONFIGURATION_FILE* `--dump-config` *NETWORK_DESCRIPTION_FILE*

This functionality can be interesting in the following cases:

- giving only one configuration file, MaBoSS completes this configuration using its defaults and generate a full configuration,

- giving several configuration files and/or configuration expressions (-e option) and/or variable definitions (-v option), MaBoSS interprets all these configurations, completes them using its defaults and generate a full configuration.

MaBoSS help

```
MaBoSS -h
MaBoSS --help

Usage:

  MaBoSS [-h|--help]

  MaBoSS [-V|--version]

  MaBoSS [-c|--config CONF_FILE] [-v|--config-vars VAR1=NUMERIC[,VAR2=...]] [-e|--config- ↩
      expr CONFIG_EXPR] -o|--output OUTPUT BOOLEAN_NETWORK_FILE

  MaBoSS [-c|--config CONF_FILE] [-v|--config-vars VAR1=NUMERIC[,VAR2=...]] [-e|--config- ↩
      expr CONFIG_EXPR] -d|--dump-config BOOLEAN_NETWORK_FILE

  MaBoSS [-c|--config CONF_FILE] [-v|--config-vars VAR1=NUMERIC[,VAR2=...]] [-e|--config- ↩
      expr CONFIG_EXPR] -l|--generate-logical-expressions BOOLEAN_NETWORK_FILE

  MaBoSS -t|--generate-config-template BOOLEAN_NETWORK_FILE

Options:

  -V --version                            : displays MaBoSS version
  -c --config CONF_FILE                   : uses CONF_FILE as a configuration file
  -v --config-vars VAR=NUMERIC[,VAR2=...] : sets the value of the given variables to the  ↩
      given numeric values
  -e --config-expr CONFIG_EXPR            : evaluates the configuration expression; may  ↩
      have multiple expressions
                                            separated by semi-colons
  -o --output OUTPUT                      : prefix to be used for output files; when  ↩
      present run MaBoSS simulation process
  -d --dump-config                        : dumps configuration and exits
  -t --generate-config-template           : generates template configuration and exits
  -l --generate-logical-expressions       : generates the logical expressions and exits
  -h --help                               : displays this message

Notices:

1. --config and --config-expr options can be used multiple times;
   multiple --config and/or --config-expr options are managed in the order given at
   the command line;
   --config-vars VAR=VALUE always overrides any VAR assignment in a configuration file or
   expression

2. --dump-config, --generate-config-template, --generate-logical-expressions and --output
   are exclusive options
```