How-To Simulate

Below are the step-by-step instructions for running simulations of stochastic Boolean regulatory networks in MATLAB, which accompanies *https://github.com/drplaugher/Math-Methods-in-Modern-Biology*.

1) Add all required paths.

2) Load model (preloaded or self-loaded)
   a. This step utilizes the `SDDS_Build(syms,f,p)` function that automatically builds the SDDS. However, users can opt to manually create data files for `F,nv,varF`. Here, `syms` are simply variables for the functions, `f`, and `p` is the number of states (usually Boolean).
   b. If users need assistance converting Boolean rules to polynomials, see the guide below.

3) Depending on model size, users can utilize MATLAB tools for Markov Chains that can:
   a. Determine attractors and fixed points
   b. Create a Transition Matrix
   c. Use a Google Matrix to achieve an approximated stationary distribution

NOTE: the state space of a BN is 2^n, which implies the transition matrix will be 2^n x 2^n

4) Under *Inductions and Control*, there are two options for inducing mutations and implementing control.
   a. **Option 1:** Alter the original functions in SDDS direction
      i. This only works for node alterations (not edges)
   b. **Option 2** for nodes: Use the *Mutation Functions* `F1-Fn` to induce the desired mutation. Users only need to update the `NODE#` and select the `ACTION` of desired knock-in (1) or knock-out (0). For example(s)

      Single mutations can be induced by using

      ```
      F1 = TruthTable_del_n_temp(F,nv,varF,p, NODE#,1);
      ```

      Doublet mutations can be induced by using

      ```
      F1 = TruthTable_del_n_temp(F,nv,varF,p, NODE#,1);

      F2 = TruthTable_del_n_temp(F1,nv,varF,p, NODE#,0);
      ```

   c. **Option 2** for edges: As with the nodes, *Mutation Functions* `F1-Fn` to induce the desired mutation. Users only need to update the `TAIL#, HEAD#`, and select the `ACTION` of desired knock-in (1) or knock-out (0). For example(s)

      ```
      F1 = TruthTable_del_a_temp(F,nv,varF,p,tail,head,v)
      ```

NOTE: Option 2 method must iterate Fi's as in the Doublet example

5) To simulate long-term trajectories of the model’s SDDS:
    a. Set the desired number of initializations and time steps.
    b. Set the desired noise level (optional) – Noisy simulations no longer have attractors.
    c. Set propensity for SDDS – typically use 0.9
    d. Make sure the function being used in the simulation matches the last function being used (aka. `F-Fn`, depending on the scenario being simulated)

NOTE: Be sure the function F in SDDS_sim matches the last used in Step 4 (if option 2)*

6) To visually represent the full trajectories of the simulation, use section *Graphing*. See that the final expression levels are those from Step 3 and should match the model’s attractor landscape.
    a. Notice that over time, expression levels will appear to converge to indicate approximated long-term states.

To determine the efficacy of controls, we compare uncontrolled simulations with the appropriate targeted control simulations. Inducing mutations will result in high levels of diseased phenotypes. Thus, a good control will produce low disease levels and high health levels (aka. apoptosis).

Convert Boolean functions to polynomials

If users need to convert their Boolean functions (rules) to polynomials (required for `SDDS_Build`), Macaulay2 is an algebraic software (web-based) that can easily achieve this goal. We have provided source code and example below.

1) Navigate to Macaulay2Web and start a new session.

2) Upload the following code (Copy/Paste). Be sure to specify your total node number `n`.

```
n = NODE#;
Xstring=apply(n,i->"x"|(i+1))
DEN=apply(Xstring,v->v|"^2-"|v)
R=ZZ/2[Xstring]/ideal(DEN/value)
RingElement | RingElement :=(x,y)->x+y+x*y;
RingElement & RingElement :=(x,y)->x*y;
```

3) Now that the environment is set, users can either convert functions individually or in bulk. For example:

```
input: x3 | x4
output: x3x4+x3+x4
--------
input: x7 & x9
output: x7x9
--------
input: ((x13+1) | x17) & x10
output: x10x13x17+x10x13+x10
```