Logical Models

First we must translate the function file to Cython.

This is not fully automated yet. First you use the script regular-expression.py.

```
python regular-expression.py INPUTFILE.txt OUTPUTFILE.txt
```

The output file should look like this.

```
vector[0] =
x[1]+2*x[1]*x[1]*x[2]+2*x[1]*x[2]*x[2]+2*x[1]*x[1]*x[2]*x[2]+2*x[1]*x[1]*x[3]+x[1]*x[1]*
=
x[18]*x[18]+x[1]+2*x[18]*x[18]*x[1]+2*x[1]*x[1]+x[18]*x[1]*x[1]+x[4]*x[4]+2*x[18]*x[18]*
=
1+x[2]*x[2]+2*x[2]*x[2]*x[4]*x[4]+2*x[5]+x[2]*x[5]+2*x[2]*x[2]*x[5]+x[5]*x[5]+2*x[2]*x[5
=
1+x[3]*x[3]+2*x[3]*x[3]*x[4]*x[4]+2*x[5]+x[3]*x[5]+2*x[3]*x[3]*x[5]+x[5]*x[5]+2*x[3]*x[5
= 1+2*x[0]+x[0]*x[0]+x[0]*x[4]+2*x[0]*x[0]*x[4]+x[4]*x[4]+2*x[0]*x[4]*x[4]
vector[5] = 1+x[5]*x[5]
vector[6] = x[14]*x[14]+x[6]+2*x[14]*x[14]*x[6]+2*x[6]*x[6]+x[14]*x[6]*x[6]
vector[7] =
x[9]*x[9]+x[12]*x[12]+2*x[9]*x[9]*x[12]*x[12]+x[7]+2*x[9]*x[9]*x[7]+2*x[12]*x[12]*x[7]+x
= 1+2*x[9]+x[9]*x[9]+x[9]*x[8]+2*x[9]*x[9]*x[8]+x[8]*x[8]+2*x[9]*x[8]*x[8]
vector[9] = x[8]+2*x[7]*x[7]*x[8]+2*x[7]*x[8]*x[8]+2*x[7]*x[7]*x[8]*x[8]
vector[10] =
x[0]+2*x[0]*x[0]*x[11]+2*x[0]*x[11]*x[11]+2*x[0]*x[0]*x[11]*x[11]+x[15]+2*x[0]*x[15]+x[0
= x[11]+2*x[11]*x[11]+x[11]*x[11]*x[12]+x[12]*x[12]+2*x[11]*x[12]*x[12]
vector[12] =
1+x[12]*x[12]+2*x[13]+x[12]*x[13]+2*x[12]*x[12]*x[13]+x[13]*x[13]+2*x[12]*x[13]*x[13]+x[
=
1+2*x[10]+x[10]*x[10]+x[10]*x[12]*x[12]+2*x[10]*x[10]*x[12]*x[12]+x[10]*x[13]+2*x[10]*x[
=
1+x[14]*x[14]+2*x[7]+x[10]*x[10]*x[7]+x[14]*x[7]+2*x[10]*x[10]*x[14]*x[7]+2*x[14]*x[14]*
=
x[14]*x[14]+x[15]+2*x[14]*x[14]*x[15]+2*x[15]*x[15]+x[14]*x[15]*x[15]+x[15]*x[15]*x[16]+
=
1+x[16]*x[16]+2*x[17]+x[16]*x[17]+2*x[16]*x[16]*x[17]+x[17]*x[17]+2*x[16]*x[17]*x[17]+x[
= x[15]*x[15]+x[17]+2*x[15]*x[15]*x[17]+2*x[17]*x[17]+x[15]*x[17]*x[17]
vector[18] = x[17]*x[17]+x[18]+2*x[17]*x[17]*x[18]+2*x[18]*x[18]+x[17]*x[18]*x[18]
vector[19] = x[19]+2*x[19]*x[19]+x[19]*x[19]*x[20]+x[20]*x[20]+2*x[19]*x[20]*x[20]
vector[20] = x[10]*x[10]+x[20]+2*x[10]*x[10]*x[20]+2*x[20]*x[20]+x[10]*x[20]*x[20]
```

All of this output file must be added to the functions.pyx for compiling.

First you must then define the function and add declare the variables.

```
@cython.boundscheck(False)
def IRP2_1(np.ndarray[DTYPE_t, ndim=1] x):
 cdef int vectorsize = x.shape[0]
 cdef np.ndarray[DTYPE_t, ndim=1] vector = np.zeros([vectorsize],dtype=int)
```

Then add everything from the output file.

```
@cython.boundscheck(False)
def IRP2_1(np.ndarray[DTYPE_t, ndim=1] x):
    cdef int vectorsize = x.shape[0]
    cdef np.ndarray[DTYPE_t, ndim=1] vector = np.zeros([vectorsize],dtype=int)
    vector[0] =
x[1]+2*x[1]*x[1]*x[2]+2*x[1]*x[2]*x[2]+2*x[1]*x[1]*x[2]*x[2]+2*x[1]*x[1]*x[3]+x[1]*x[1]*
vector[1] =
x[18]*x[18]+x[1]+2*x[18]*x[18]*x[1]+2*x[1]*x[1]+x[18]*x[1]*x[1]+x[4]*x[4]+2*x[18]*x[18]*
vector[2] =
1+x[2]*x[2]+2*x[2]*x[2]*x[4]*x[4]+2*x[5]+x[2]*x[5]+2*x[2]*x[2]*x[5]+x[5]*x[5]+2*x[2]*x[5
vector[3] =
1+x[3]*x[3]+2*x[3]*x[3]*x[4]*x[4]+2*x[5]+x[3]*x[5]+2*x[3]*x[3]*x[5]+x[5]*x[5]+2*x[3]*x[5
vector[4] = 1+2*x[0]+x[0]*x[0]+x[0]*x[4]+2*x[0]*x[0]*x[4]+x[4]*x[4]+2*x[0]*x[4]*x[4]
    vector[5] = 1+x[5]*x[5]
    vector[6] = x[14]*x[14]+x[6]+2*x[14]*x[14]*x[6]+2*x[6]*x[6]+x[14]*x[6]*x[6]
    vector[7] =
x[9]*x[9]+x[12]*x[12]+2*x[9]*x[9]*x[12]*x[12]+x[7]+2*x[9]*x[9]*x[7]+2*x[12]*x[12]*x[7]+x
vector[8] = 1+2*x[9]+x[9]*x[9]+x[9]*x[8]+2*x[9]*x[9]*x[8]+x[8]*x[8]+2*x[9]*x[8]*x[8]
    vector[9] = x[8]+2*x[7]*x[7]*x[8]+2*x[7]*x[8]*x[8]+2*x[7]*x[7]*x[8]*x[8]
    vector[10] =
x[0]+2*x[0]*x[0]*x[11]+2*x[0]*x[11]*x[11]+2*x[0]*x[0]*x[11]*x[11]+x[15]+2*x[0]*x[15]+x[0
vector[11] = x[11]+2*x[11]*x[11]+x[11]*x[11]*x[12]+x[12]*x[12]+2*x[11]*x[12]*x[12]
    vector[12] =
1+x[12]*x[12]+2*x[13]+x[12]*x[13]+2*x[12]*x[12]*x[13]+x[13]*x[13]+2*x[12]*x[13]*x[13]+x[
vector[13] =
1+2*x[10]+x[10]*x[10]+x[10]*x[12]*x[12]+2*x[10]*x[10]*x[12]*x[12]+x[10]*x[13]+2*x[10]*x[
vector[14] =
1+x[14]*x[14]+2*x[7]+x[10]*x[10]*x[7]+x[14]*x[7]+2*x[10]*x[10]*x[14]*x[7]+2*x[14]*x[14]*
vector[15] =
x[14]*x[14]+x[15]+2*x[14]*x[14]*x[15]+2*x[15]*x[15]+x[14]*x[15]*x[15]+x[15]*x[15]*x[16]+
vector[16] =
1+x[16]*x[16]+2*x[17]+x[16]*x[17]+2*x[16]*x[16]*x[17]+x[17]*x[17]+2*x[16]*x[17]*x[17]+x[
vector[17] = x[15]*x[15]+x[17]+2*x[15]*x[15]*x[17]+2*x[17]*x[17]+x[15]*x[17]*x[17]
    vector[18] = x[17]*x[17]+x[18]+2*x[17]*x[17]*x[18]+2*x[18]*x[18]+x[17]*x[18]*x[18]
    vector[19] = x[19]+2*x[19]*x[19]+x[19]*x[19]*x[20]+x[20]*x[20]+2*x[19]*x[20]*x[20]
    vector[20] = x[10]*x[10]+x[20]+2*x[10]*x[10]*x[20]+2*x[20]*x[20]+x[10]*x[20]*x[20]
    for i in xrange(0,len(vector)):
        while vector[i] > 3:
            vector[i] = vector[i]%3
    return vector
```

And save this file and exit it.

Compiling

We must now compile the cython code.

```
cython functions.pyx
gcc -shared -pthread -fPIC -fwrapv -O2 -Wall -fno-strict-aliasing
-I/usr/include/python2.7 -o functions.so functions.c
```

Running on single processor computer

Now that the code is compiled, we can run it. The main program is called main_attractor_synch_cython.py

Line 19 of the code determines which function to run. This is the name that you defined in above (IRP2_1).

Change line 19 to

```
from functions import IRP2_1 as function
```

Save.

From command line

```
python main_attractor_synch_cython.py \-h

OUTPUT should be
optional arguments:
  -h, --help            show this help message and exit
  -n NUMBERSTATES, --numberstates NUMBERSTATES
                        provide a number of states
  -nn NUMBERNODES, --numbernodes NUMBERNODES
                        provide a number of nodes
  -s START, --start START
                        starting string to convert to base Nstates
  -e END, --end END     ending string to convert to based Nstates
  -v VERBOSE, --verbose VERBOSE
                        if you want verbose updates (use with single
                        processor)
```

Thus we would run the program

```
python main_attractor_synch_cython.py -n 3 -nn 21 -s 0 -e 1000 -v
```

We could split this up on a 100s of cores on a cluster.

Running on multiple processors/across nodes

To run on a multi core or cluster, you need to comment the main() (line 105) and uncomment all the multiprocessing lines below that. In python it is just the two lines that contain """, line 111 and 191.
Then you run using mpirun

```
mpirun -n #proc /path/to/python main_attractor_synch_cython.py -n 3 -nn 21 -s 0 -e
1000
```