The constructor for a gate:

#Gate(name,Km,n,gateType,mB,pB,m\_halfLife, p\_halfLife)

I finished cleaning up the code and started to look into converting a netlist (as opposed to a circuit string) into a dag. This will allow for a more universal way to convert. I paused though to look back at the problem I was having yesterday where the libraries were not working for generating the graphs.

I spoke to Tom about the problem with the nan and the odeint not working with the randomly generated libraries and we found that if we lowered the input range it worked. We lowered the input range from [0,100] to [0,10] and all the graphs were able to successfully finish. Apparently ODEINT was having a problem with how dramatically the input values changed since they were step functions.

Now I am moving back to going from the netlist to the dag. First I will make a program that can split the netlists Bryan sent me into individual files for each truth value that only contain the netlist.

I was having a problem with converting the json file to a json object. It worked for the file with the OR but not with the other file for some reason. I can’t figure out why so I will need to do a more manual method for converting the file.

I noticed that the wrapper function messes up for making a circuit out of “(a.0)” so I paused again to look into this. I think it is a problem with odeint. The problem was with odeint. The fix was to set a list of critical times for odeint to be wary of when calculating.

Now I will move back to splitting up the netlist.

I finished splitting the netlist and finished making a program that can go from the netlist to the graphs. It runs into problems with sequential circuits. I need to look into this.

It seems to be working now except I keep getting a math domain error.

It was a problem with having the ymax lower than ymin. I fixed this problem.