CONDITIONS FOR ADDING THE PHOSPHORYLATION STATES:

* Can we remove the entries which do not have a gene name associated with it?
* Also there are keys naming 1-Sep (Dates)etc. What to do with it?
* 1x or Mono Phosphorylations
  1. Replacing the 1xS or 1xT types of Nodes by performing regular expression checks. It is replaced by the 1xS11 sort of thing.
     + ***if (re.match('1xS\d|1xT\d|1xS/T', ele) is None) and '1x' in ele***it is added to
     + ***if (re.match('1xS\d|1xT\d',str(ele1)) is not None) and '1x' in str(ele1):***
  2. 1xS/T are not touched and are kept as it is for now. The reason is we have 1xS/T, 1xS17, 1xS12 ,suppose, then to which should we add to. S17 and S12 already have none of their values as zeroes
* 2x or Di Phosphorylations
  1. Replacing 2xS/T-S11 types of Nodes by splitting the strings and the checking that if the S11 is available in other di phosphorylation sites. If it is there it is added.
     + **if 'S/T' in ele and '2x' in ele and '2xS/T' not in ele:**
     + **if (re.match('S\d|T\d',value) is not None):**
     + **ele2 =value**
     + **if ele2 in ele1 and '2x' in ele1 and 'S/T' not in ele1:**
  2. 2xS/T are not touched
* 3x or Tri Phosphorylations
  1. Replacing 3xS/T-S12-S11 types of Nodes by splitting the string and creating a set of (S12,S11). Then checking if this is the subset of any other tri phosphorylation. If yes then they are added.
     + **u = set (t.split('-').remove('S/T'))**
     + **u1 = set (t1.split('-'))**
     + **if u.issubset(u1):**
  2. 3xPhospho [S14(100); T/S] which has two undetermined positions is stored as 3xS14-S/T. This will be added to any tri phosphorylation one which has S14 in it.
  3. 3xS/T are not touched

REWIRING NETWORK ALGORITHM

Regarding rewiring I wasn't able to reduce it. Following is what I did:

1. If mono exist
   1. All mono to source
   2. If di exist
      1. Check mono in di
         1. If yes check tri exist
            1. If tri exist check that di is in tri

If yes the edge from tri to finish

* + - * 1. If no: edge from di to tri
      1. If no: then edge from di to tri
    1. If mono is not in di then edge from mono to finish
  1. If no di but tri exist
     1. Check mono in tri
        1. If exist edge from tri to finish
     2. If not in tri then mono to finish
  2. If no di tri exist
     1. mono to finish

1. If no mono but di exist
   1. All di to source
   2. Check tri exist
      1. Check di in tri
         1. If yes then tri to finish
      2. If no the di to finish
   3. If tri doesn't exist then di to finish
2. If no mono and di exist
   1. Add tri to source
   2. Add edge tri to finish