1 Simulate 3.e with Viterbi Algorithm

In [1]:

```
states = ('Happy', 'Angry')
observations = ('smile', 'frown', 'laugh', 'yell')
transition_probability = {
   'Happy' : {'Happy': 0.9, 'Angry': 0.1},
   'Angry' :   {'Happy': 0.1, 'Angry': 0.9},
   }
emission_probability = {
   'Happy' : {'smile': 0.6, 'frown': 0.1, 'laugh': 0.2, 'yell':0.1},
   'Angry'   : {'smile': 0.1, 'frown': 0.6, 'laugh': 0.1,  'yell': 0.2},
   }

def Viterbit(obs, states, s_pro, t_pro, e_pro):
    path = { s:[] for s in states} # init path: path[s] represents the path ends with s
    curr_pro = {}
    for s in states:
        curr_pro[s] = s_pro[s]*e_pro[s][obs[0]]
    for i in range(1, len(obs)):
        last_pro = curr_pro
        curr_pro = {}
        for curr_state in states:
            max_pro, last_sta = max(((last_pro[last_state]*t_pro[last_state][curr_state]*e_pro[curr_
                                     for last_state in states))
            curr_pro[curr_state] = max_pro
            path[curr_state].append(last_sta)

    # find the final largest probability
    max_pro = -1
    max_path = None
    for s in states:
        path[s].append(s)
        if curr_pro[s] > max_pro:
            max_path = path[s]
            max_pro = curr_pro[s]
    return max_path


if __name__ == '__main__':
    obs = ['frown', 'frown', 'frown','frown','frown']
    # Test for the threshold
    p_list = [54/55 + 1e-5, 54/55 - 1e-5]
    for p in p_list:
        start_probability = {'Happy': p, 'Angry': 1-p}
        print(start_probability)
        print (Viterbit(obs, states, start_probability, transition_probability, emission_probability
```

executed in 20ms, finished 23:23:42 2019-09-15

```
{'Happy': 0.9818281818181818, 'Angry': 0.018171818181818233}
['Happy', 'Angry', 'Angry', 'Angry', 'Angry']
{'Happy': 0.9818081818181819, 'Angry': 0.018191818181818142}
['Angry', 'Angry', 'Angry', 'Angry', 'Angry']
```

In [ ]: