```
%%% Get probabilities of settling into FPs from all initial states with unique
first update orders

% Function to find final states (fixed points) starting from all initial states
and probabilities of all transitions
function [transitions, FPdictionary] = absorptionProbabilities(T,
binaryStatesCell, allOrders, statesDictionary)

% Display node numbers and corresponding states
disp('Numeric states and corresponding node numbers:')
disp(statesDictionary);

transitions = {}; % Initialize cell to store initial state, final state, and
transition probabilities

FPs = []; % Initialize vector to store FPs

FPindices = []; % Initialize vector to store FP state indices (graph-node ID)

FPdictionary = dictionary(FPs, FPindices); % FPs as keys and indices as values

% Nested loops to find transition probabilities
for initialState = 1:length(binaryStatesCell) % Iterate through all initial
states

    for inOrders = 1:length(allOrders) % Iterate through all update orders

        chosenOrder = allOrders(inOrders, :); % Extract current update order

        sii = binaryStatesCell{initialState}; % Extract initial state

        si = sii; % Initialize transient state

        sj = si; % Initialize final state

        count = 1; % Record number of iterations before FP is reached

        for iterations = 1:100 % 99 iterations considered after 1st one with
determined update order because number of states before FP is reached would be
lesser than 100

            sii = si; % Set initial state as previous iteration's final state

            if count ~= 1 % For every iteration except the first one

                chosenOrder = allOrders(randi([1, 720]), :); % Choose random
update order

            end

            for inOrder = 1:6 % For every position in chosen update order (for
every interval)
```

```
                  if chosenOrder(inOrder) == 1
                      % f1(v2, v5: T = 1) = NOT(v2) OR NOT(v5)
                      condition_1 = si(2) == '0' || si(5) == '0';

                      if condition_1 == 1
                          sj(1) = '1';
                      elseif condition_1 == 0
                          sj(1) = '0';
                      end
                      si = sj;

                  elseif chosenOrder(inOrder) == 2
                      % f2(v3, v4: T = 1) = NOT(v3) OR NOT(v4)
                      condition_2 = si(3) == '0' || si(4) == '0';

                      if condition_2 == 1
                          sj(2) = '1';
                      elseif condition_2 == 0
                          sj(2) = '0';
                      end
                      si = sj;

                  elseif chosenOrder(inOrder) == 3
                      % f3(v1, v6: T = 3) = v1 AND NOT(v6)
                      condition_3 = si(1) == '1' && si(6) == '0';

                      if condition_3 == 1
                          sj(3) = '1';
                      elseif condition_3 == 0
                          sj(3) = '0';
                      end
                      si = sj;

                  elseif chosenOrder(inOrder) == 4
                      % f4(v3, v2, v5: T = 3) = v3 AND NOT(v2) OR NOT(v5)
                      condition_4 = (si(3) == '1' && si(2) == '0') || si(5) == '0';

                      if condition_4 == 1
                          sj(4) = '1';
                      elseif condition_4 == 0
                          sj(4) = '0';
                      end
                      si = sj;

                  elseif chosenOrder(inOrder) == 5
                      % f5(v4: T = 2) = NOT(v4)
                      condition_5 = si(4) == '0';

                      if condition_5 == 1
                          sj(5) = '1';
                      elseif condition_5 == 0
                          sj(5) = '0';
                      end
                      si = sj;
```

```matlab
                elseif chosenOrder(inOrder) == 6
                    % f6(v3, v4: T = 2) = NOT(v3) OR NOT(v4)
                    condition_6 = si(3) == '0' || si(4) == '0';

                    if condition_6 == 1
                        sj(6) = '1';
                    elseif condition_6 == 0
                        sj(6) = '0';
                    end
                    si = sj;

                end

            end

            count = count + 1; % Increment by 1 after 1 iteration is completed
(network-state changed)

            sjNumeric = str2num(sj); % Convert final state to numeric to use for
iterative probability calculation

            % Iterative probability calculation
            T_ij = T(initialState, statesDictionary(sjNumeric)); % Extract STM
transition probability value

            if iterations == 1 % For first iteration

                p = T_ij; % No p0

                p0 = p; % Assign value to p0

            elseif iterations ~= 1 % For iterations except the first

                p = T_ij * p0; % Product of p0 and transition probability value

                p0 = p; % Update p0

            end

            % Find FP by checking for same initial and final states
            if sii == sj % Condition favouring an FP

                transitions{end + 1, 1} = binaryStatesCell{initialState}; %
Initial state

                transitions{end, 2} = sj; % FP

                transitions{end, 3} = p; % Transition probability

                break % Go to new update order

            end
```

```
        end

        % Add entries to FP dictionary
        if isKey(FPdictionary, sjNumeric) % Check for presence of FP final state

            continue; % Go to next update order

        else

            FPdictionary(sjNumeric) = statesDictionary(sjNumeric); % FP found
and add entry

        end

    end

end

% Display all fixed points
disp('FPs:')
disp(FPdictionary);

end

% Store all transitions and their probabilities
[transitions, fp_dict] = absorptionProbabilities(T, binaryStatesCell, allOrders,
statesDictionary);
```

Output:

```
Numeric states and corresponding node numbers:

0       □ 1

1       □ 2

10      □ 3

11      □ 4

100     □ 5

101     □ 6

110     □ 7

111     □ 8

1000    □ 9

1001    □ 10

1010    □ 11

1011    □ 12

1100    □ 13
```

```
1101   ⍰ 14
1110   ⍰ 15
1111   ⍰ 16
10000  ⍰ 17
10001  ⍰ 18
10010  ⍰ 19
10011  ⍰ 20
10100  ⍰ 21
10101  ⍰ 22
10110  ⍰ 23
10111  ⍰ 24
11000  ⍰ 25
11001  ⍰ 26
11010  ⍰ 27
11011  ⍰ 28
11100  ⍰ 29
11101  ⍰ 30
11110  ⍰ 31
11111  ⍰ 32
100000 ⍰ 33
100001 ⍰ 34
100010 ⍰ 35
100011 ⍰ 36
100100 ⍰ 37
100101 ⍰ 38
100110 ⍰ 39
100111 ⍰ 40
101000 ⍰ 41
101001 ⍰ 42
101010 ⍰ 43
101011 ⍰ 44
101100 ⍰ 45
```

```
101101 ▯ 46

101110 ▯ 47

101111 ▯ 48

110000 ▯ 49

110001 ▯ 50

110010 ▯ 51

110011 ▯ 52

110100 ▯ 53

110101 ▯ 54

110110 ▯ 55

110111 ▯ 56

111000 ▯ 57

111001 ▯ 58

111010 ▯ 59

111011 ▯ 60

111100 ▯ 61

111101 ▯ 62

111110 ▯ 63

111111 ▯ 64


FPs:

10011  ▯ 20

110101 ▯ 54

101100 ▯ 45
```