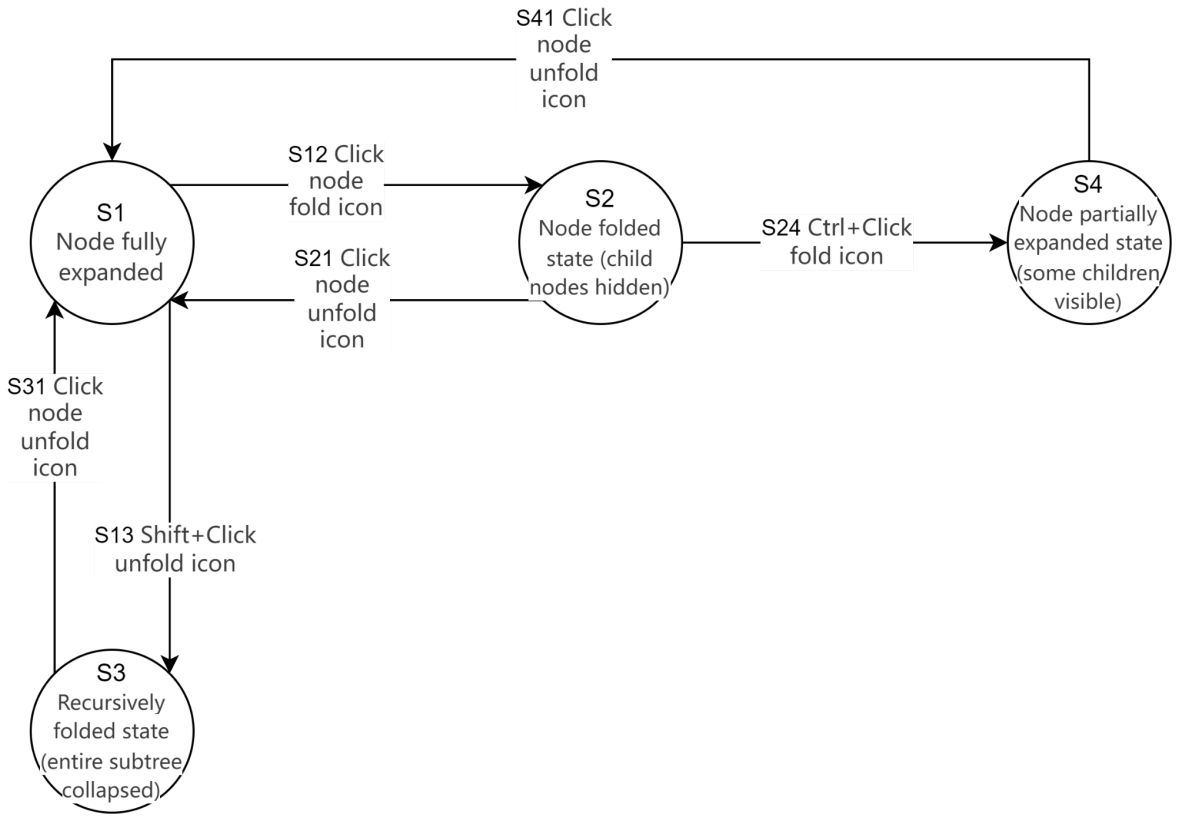
11-Markov_chain

Function1

FSM



Log File

```
2025-05-10 17:00:16.044,testUser,S1,E1,S2
2025-05-10 17:00:17.218,testUser,S2,E2,S1
2025-05-10 17:00:25.994,testUser,S1,E1,S2
2025-05-10 17:00:26.650,testUser,S2,E4,S3
2025-05-10 17:00:27.171,testUser,S2,E4,S3
2025-05-10 17:00:27.954,testUser,S1,E1,S2
2025-05-10 17:00:29.123,testUser,S2,E4,S3
2025-05-10 17:00:29.818,testUser,S2,E4,S3
2025-05-10 17:00:33.812,testUser,S1,E1,S2
2025-05-10 17:00:37.500,testUser,S2,E2,S1
2025-05-10 17:00:37.504,testUser,S4,E2,S1
2025-05-10 17:00:37.505,testUser,S4,E2,S1
2025-05-10 17:00:37.505,testUser,S4,E2,S1
2025-05-10 17:00:37.506,testUser,S4,E2,S1
2025-05-10 17:00:37.507,testUser,S4,E2,S1
2025-05-10 17:00:37.508,testUser,S4,E2,S1
2025-05-10 17:00:37.508,testUser,S4,E2,S1
2025-05-10 17:00:37.510,testUser,S4,E2,S1
2025-05-10 17:00:38.458,testUser,S1,E1,S2
2025-05-10 17:00:38.464,testUser,S1,E3,S4
2025-05-10 17:00:38.465,testUser,S1,E3,S4
2025-05-10 17:00:38.466,testUser,S1,E3,S4
2025-05-10 17:00:38.467,testUser,S1,E1,S2
2025-05-10 17:00:38.468,testUser,S1,E3,S4
2025-05-10 17:00:38.468,testUser,S1,E3,S4
2025-05-10 17:00:38.469,testUser,S1,E1,S2
2025-05-10 17:00:38.470,testUser,S1,E3,S4
2025-05-10 17:00:38.470,testUser,S1,E3,S4
2025-05-10 17:00:38.471,testUser,S1,E3,S4
2025-05-10 17:01:05.964,testUser,S2,E4,S3
2025-05-10 17:01:06.675,testUser,S2,E4,S3
2025-05-10 17:01:07.627,testUser,S2,E4,S3
2025-05-10 17:01:14.092,testUser,S1,E1,S2
2025-05-10 17:01:15.146,testUser,S1,E1,S2
2025-05-10 17:02:01.067,testUser,S2,E2,S1
2025-05-10 17:02:02.203,testUser,S2,E2,S1
2025-05-10 17:02:05.882,testUser,S4,E2,S1
2025-05-10 17:02:05.883,testUser,S4,E2,S1
2025-05-10 17:02:05.883,testUser,S4,E2,S1
2025-05-10 17:02:05.884,testUser,S4,E2,S1
2025-05-10 17:02:05.884,testUser,S3,E2,S1
2025-05-10 17:02:05.885,testUser,S4,E2,S1
2025-05-10 17:02:05.885,testUser,S4,E2,S1
2025-05-10 17:02:05.885,testUser,S4,E2,S1
2025-05-10 17:02:05.886,testUser,S2,E2,S1
2025-05-10 17:02:05.887,testUser,S4,E2,S1
2025-05-10 17:02:05.888,testUser,S4,E2,S1
2025-05-10 17:02:05.888,testUser,S2,E2,S1
```

```
2025-05-10 17:02:05.890,testUser,S4,E2,S1
2025-05-10 17:02:05.891,testUser,S4,E2,S1
2025-05-10 17:02:05.892,testUser,S4,E2,S1
2025-05-10 17:02:16.131,testUser,S1,E1,S2
2025-05-10 17:02:19.099,testUser,S2,E2,S1
2025-05-10 17:02:23.914,testUser,S1,E1,S2
2025-05-10 17:02:24.476,testUser,S2,E4,S3
2025-05-10 17:02:25.434,testUser,S2,E4,S3
2025-05-10 17:02:25.914,testUser,S2,E4,S3
```

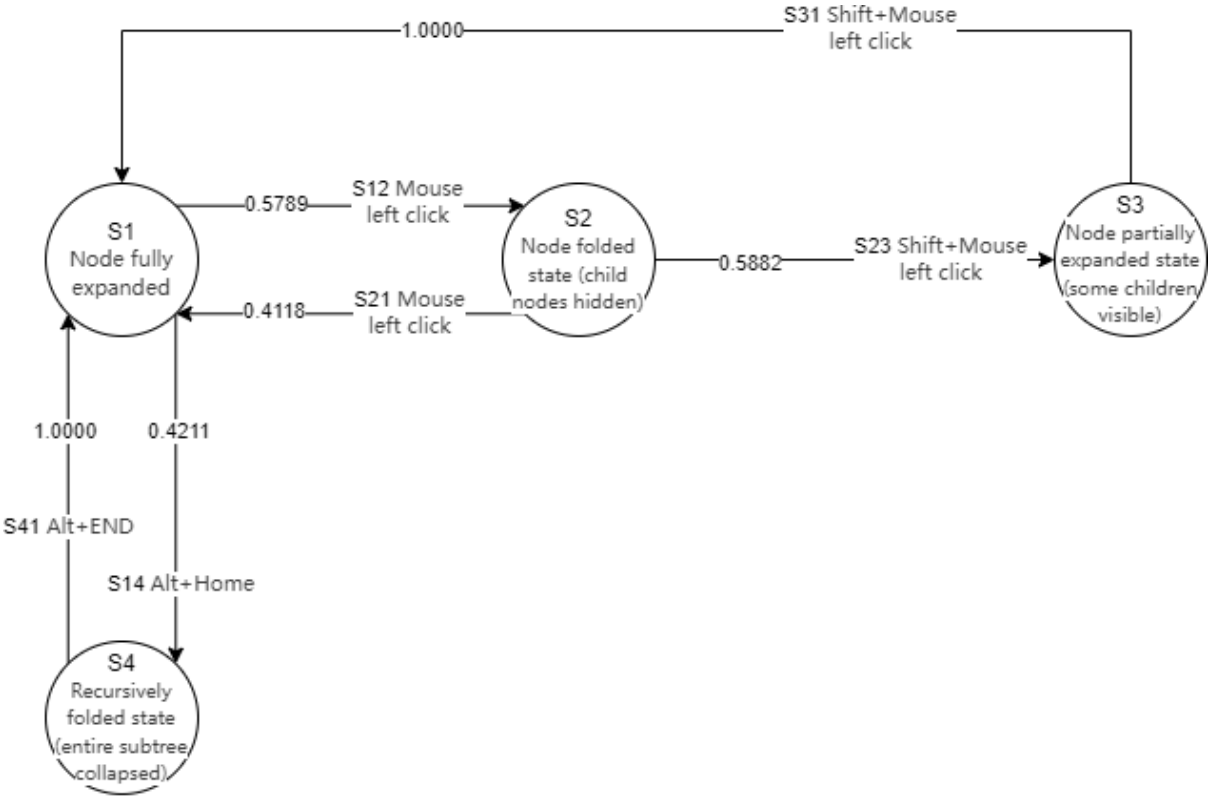
Probability Calculation Code

freeplane > analyze_markov_chain.py > ...

```python
# analyze_markov_chain.py
from collections import defaultdict

log_file = r'BIN\node_fsm.log'

transition_counts = defaultdict(lambda: defaultdict(int))
state_counts = defaultdict(int)

with open(log_file, encoding='utf-8') as f:
    for line in f:
        parts = line.strip().split(',')
        # 跳过空行或格式不对的行
        if len(parts) < 5:
            continue
        # 取 from_state 和 to_state
        from_state = parts[2]
        to_state = parts[4]
        transition_counts[from_state][to_state] += 1
        state_counts[from_state] += 1

print("From_State,To_State,Count,Probability")
for from_state, to_states in transition_counts.items():
    total = state_counts[from_state]
    for to_state, count in to_states.items():
        prob = count / total if total > 0 else 0
        print(f"{from_state},{to_state},{count},{prob:.4f}")
```

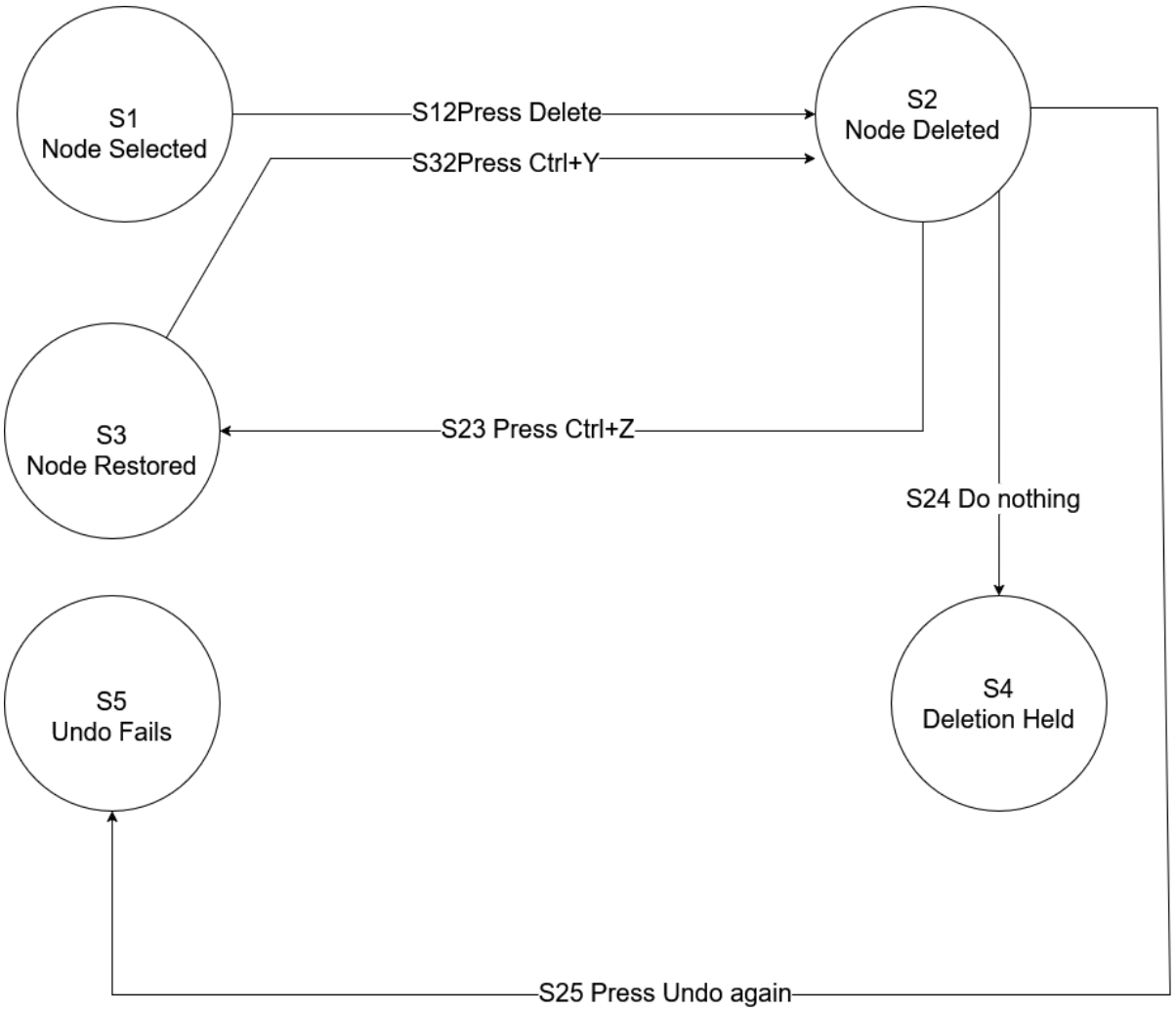
The Markov Chain

```
PS E:\5620_5618\FP\freeplane> python analyze_markov_chain.py
From_State,To_State,Count,Probability
S1,S2,11,0.5789
S1,S4,8,0.4211
S2,S1,7,0.4118
S2,S3,10,0.5882
S4,S1,20,1.0000
S3,S1,1,1.0000
```



Function 2

FSM



Log File

```
2025-05-12 16:25:13.928,testUser,S1,E1,S2
2025-05-12 16:25:13.933,testUser,S2,E2,S3
2025-05-12 16:25:13.933,testUser,S3,E3,S2
2025-05-12 16:25:13.934,testUser,S2,E4,S4
2025-05-12 16:25:13.935,testUser,S2,E5,S5
2025-05-12 16:32:12.816,testUser,S1,E1,S2
2025-05-12 16:32:12.820,testUser,S2,E2,S3
2025-05-12 16:32:12.821,testUser,S3,E3,S2
2025-05-12 16:32:12.822,testUser,S2,E4,S4
2025-05-12 16:32:12.822,testUser,S2,E5,S5
2025-05-12 16:32:57.502,testUser,S1,E1,S2
2025-05-12 16:32:57.507,testUser,S2,E2,S3
2025-05-12 16:32:57.507,testUser,S3,E3,S2
2025-05-12 16:32:57.508,testUser,S2,E4,S4
2025-05-12 16:32:57.509,testUser,S2,E5,S5
2025-05-12 16:44:40.918,testUser,S1,E1,S2
2025-05-12 16:44:40.934,testUser,S2,E2,S3
2025-05-12 16:44:40.935,testUser,S3,E3,S2
2025-05-12 16:44:40.935,testUser,S2,E4,S4
2025-05-12 16:44:40.936,testUser,S2,E5,S5
```

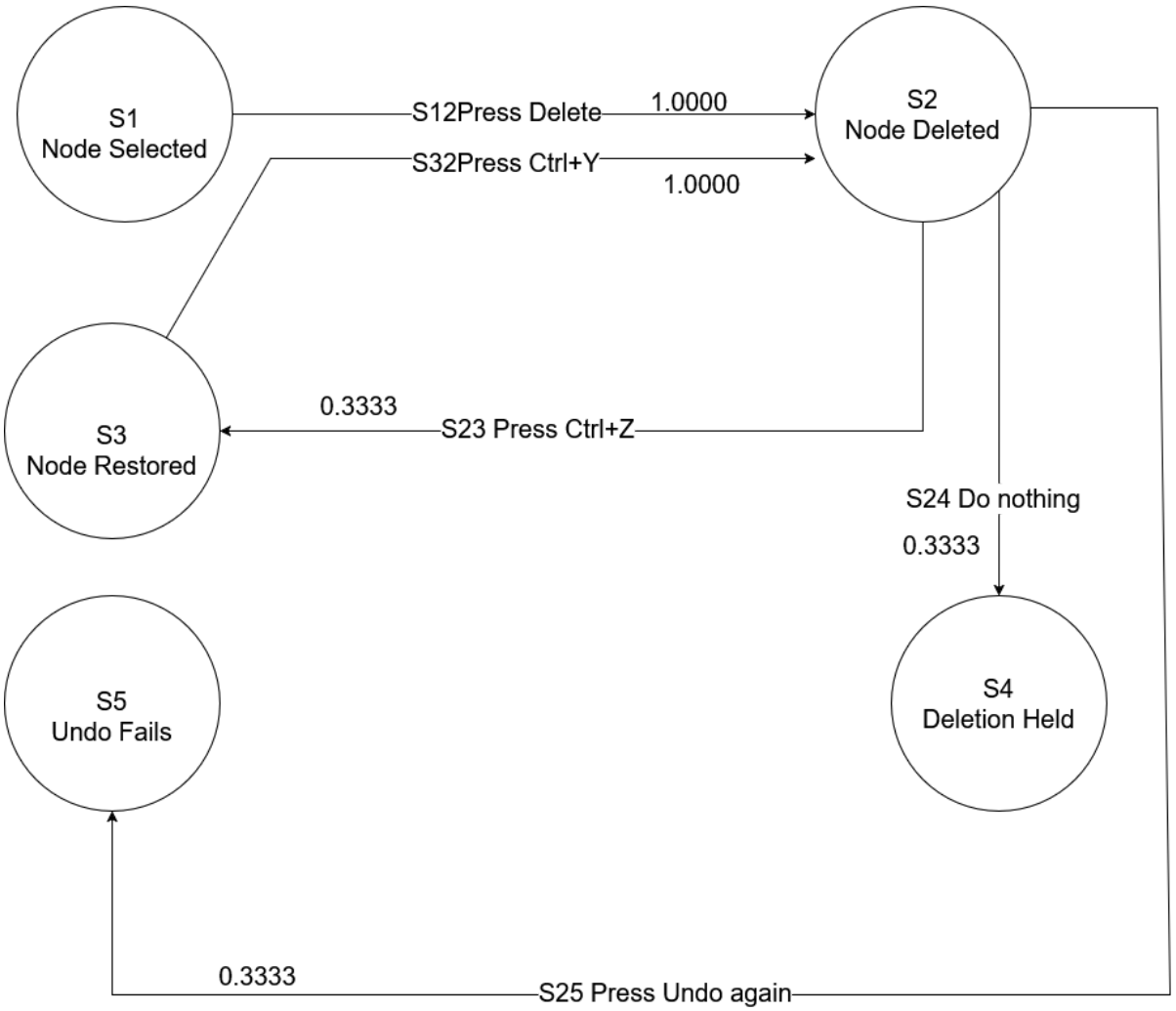
Probability Calculation Code

MarkovProbabilityCode(function2).py

```python
log_file = 'markov_log.csv'

# Dictionary structure: {start_state: [list of end_states]}
start = {}

with open(log_file, 'r') as f:
    for line in f:
        parts = line.strip().split(",")
        if len(parts) != 5:
            continue
        from_state = parts[2]
        to_state = parts[4]

        if from_state in start:
            start[from_state].append(to_state)
        else:
            start[from_state] = [to_state]

# Print transition probabilities
print("Transition Probabilities:\n")
for key in start:
    ends = start[key]
    searched = []
    for end in ends:
        if end not in searched:
            prob = ends.count(end) / len(ends)
            print(f"{key} => {end} : {prob:.4f}")
            searched.append(end)

```

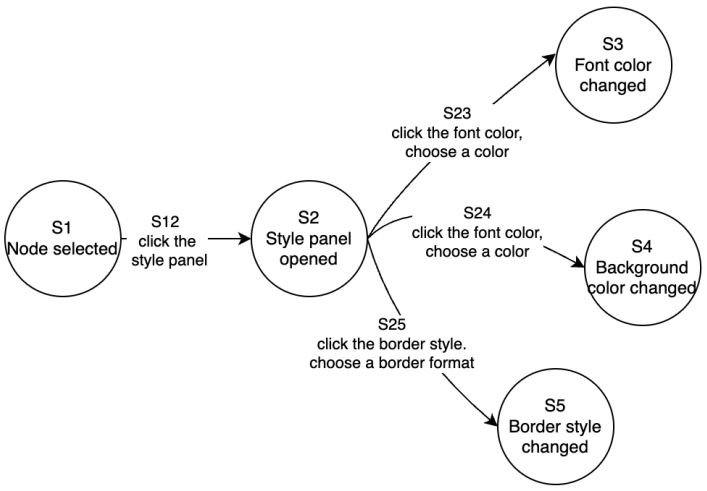
The Markov Chain

```
S1 => S2 : 1.0000
S2 => S3 : 0.3333
S2 => S4 : 0.3333
S2 => S5 : 0.3333
S3 => S2 : 1.0000
```


S1
Node Selected
S12Press Delete
1.0000
S2
Node Deleted
S32Press Ctrl+Y
1.0000
S3
Node Restored
0.3333
S23 Press Ctrl+Z
S24 Do nothing
0.3333
S5
Undo Fails
S4
Deletion Held
0.3333
S25 Press Undo again

Function 3

FSM



Log File

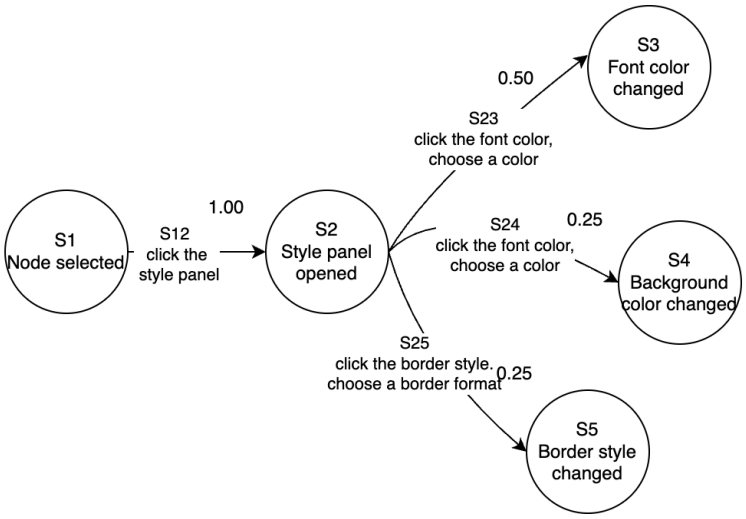
[14/May/2025 01:29:14] S1→S2
[14/May/2025 01:29:43] S2→S3
[14/May/2025 01:30:49] S2→S5
[14/May/2025 01:31:18] S2→S3
[14/May/2025 01:32:12] S2→S4

Probability Calculation Code

```
S1→S2: 1/1 = 1.00
S2→S3: 2/4 = 0.50
S2→S4: 1/4 = 0.25
S2→S5: 1/4 = 0.25
```

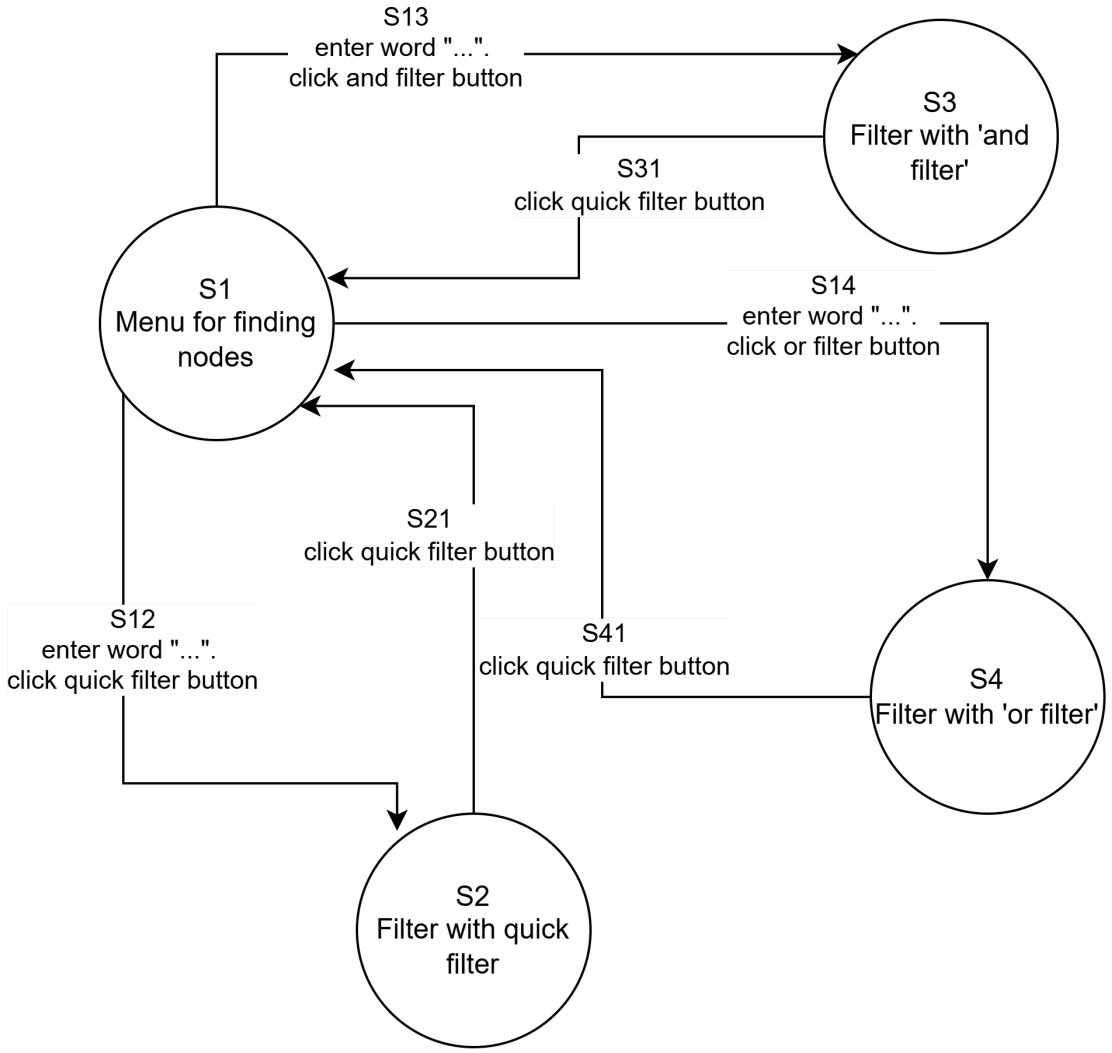
```python
import re
from collections import defaultdict

def parse_log_file(filepath):
    transition_counts = defaultdict(int)
    from_state_counts = defaultdict(int)

    pattern = re.compile(r'\[\d{2}/[A-Za-z]{3}/\d{4} \d{2}:\d{2}:\d{2}] (\S+)→(\S

    with open(filepath, 'r', encoding='utf-8') as f:
        for line in f:
            match = pattern.search(line)
            if match:
                from_state, to_state = match.groups()
                transition = f"{from_state}→{to_state}"
                transition_counts[transition] += 1
                from_state_counts[from_state] += 1

    return transition_counts, from_state_counts

def compute_probabilities(transition_counts, from_state_counts):
    probabilities = {}
    for transition, count in transition_counts.items():
        from_state = transition.split("→")[0]
        total = from_state_counts[from_state]
        probabilities[transition] = count / total
    return probabilities

def print_analysis(transition_counts, from_state_counts, probabilities):
    print("FSM Transition Probability Analysis\n")
    for transition in sorted(transition_counts.keys()):
        count = transition_counts[transition]
        from_state = transition.split("→")[0]
        total = from_state_counts[from_state]
        probability = probabilities[transition]
        print(f"{transition}: {count}/{total} = {probability:.2f}")

if __name__ == "__main__":
    filepath = "fsm_log.txt"
    transition_counts, from_state_counts = parse_log_file(filepath)
    probabilities = compute_probabilities(transition_counts, from_state_counts)
    print_analysis(transition_counts, from_state_counts, probabilities)
```

The Markov Chain


S1
Node selected
1.00
S12
click the
style panel
S2
Style panel
opened
0.50
S23
click the font color,
choose a color
S3
Font color
changed
S24
click the font color,
choose a color
0.25
S4
Background
color changed
S25
click the border style.
choose a border format
0.25
S5
Border style
changed

Function 4

FSM



Log File

```
S1,S2,S12
S2,S1,S21
S1,S3,S13
S3,S1,S31
S1,S4,S14
S4,S1,S41
```

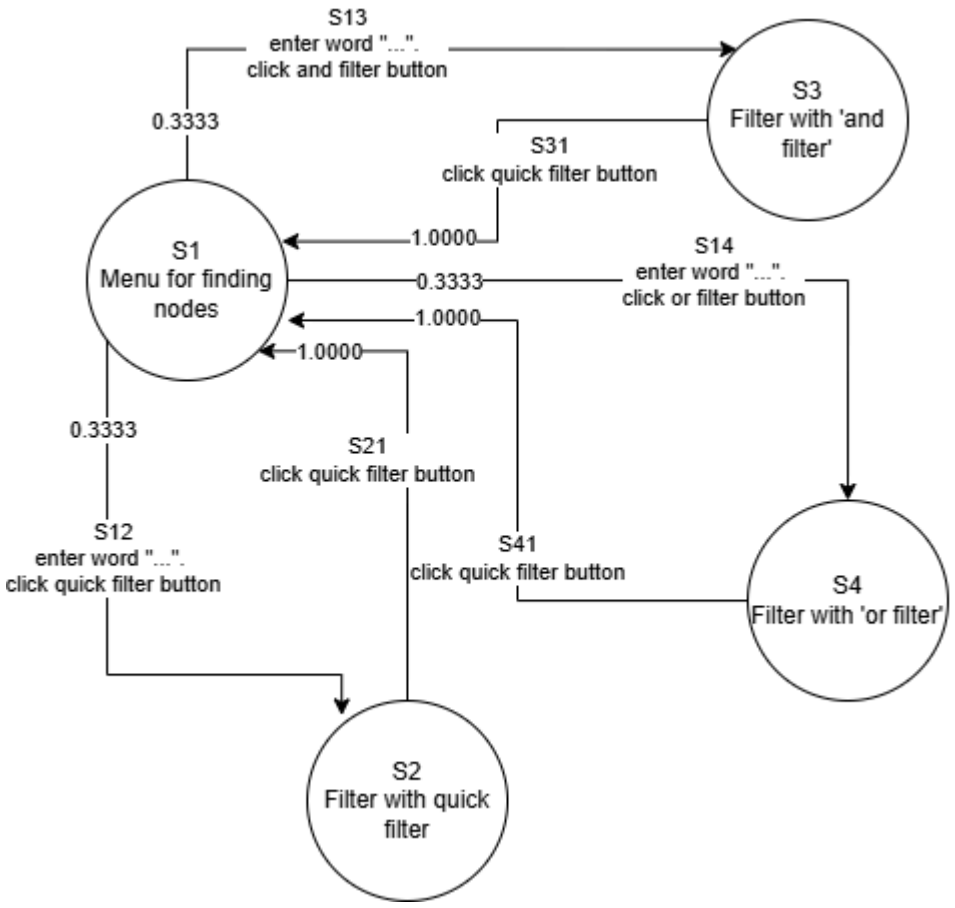
System.out.println outputs to the terminal, using test cases to test.

Probability Calculation Code

```
log_file = 'MarkovLog.txt'

# open file and sort movement of states in dictionary with start as key
start = {}
with open(log_file) as f:
    for line in f:
        path = line.split(",")
        if path[0] in start:
            start[path[0]].append(path[1])
        else:
            start[path[0]] = [path[1]]

# print results
print(start)
for key in start:
    ends = start[key]
    searched = []
    for end in ends:
        if end not in searched:
            print(key, "=>", end, ": {:.4f}".format(ends.count(end)/len(ends)))

```

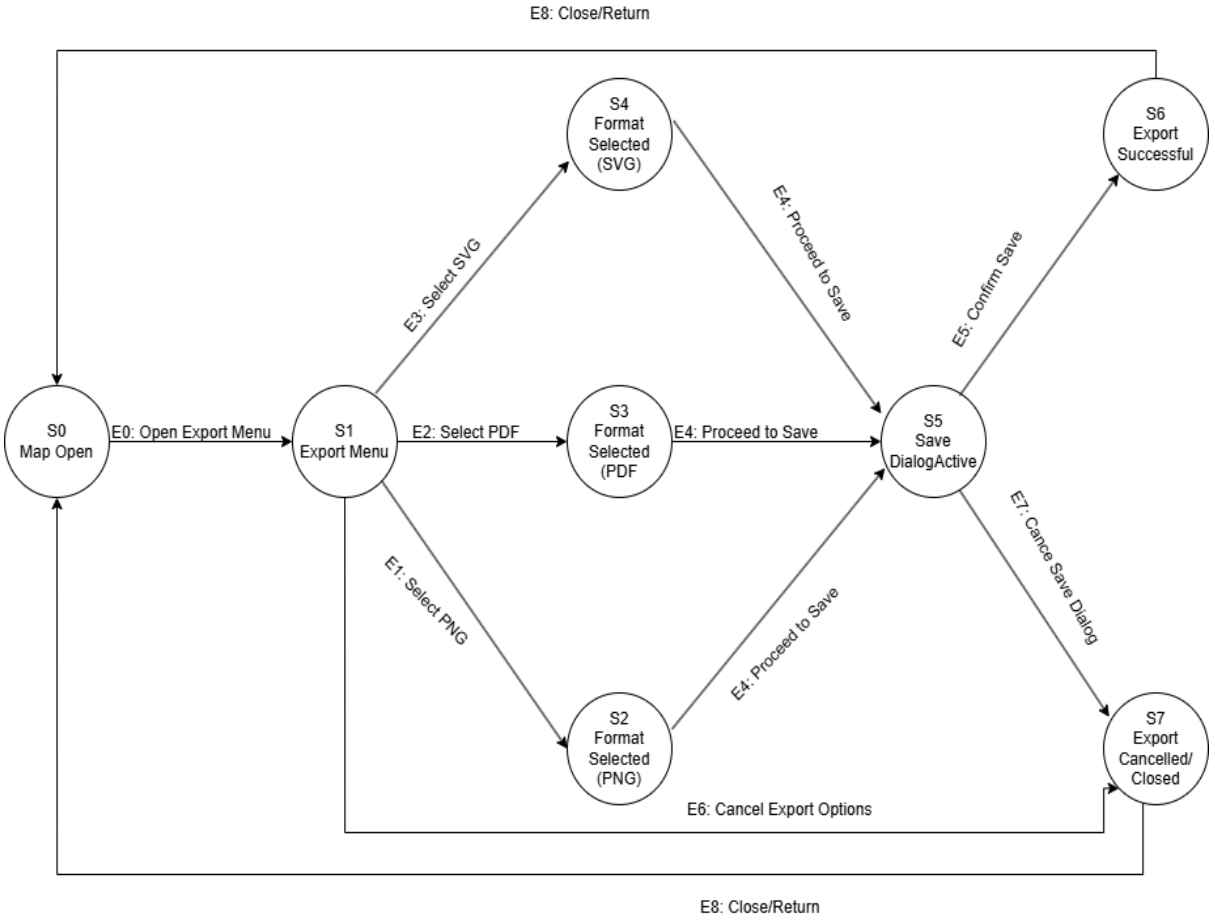
The Markov Chain

```
{'S1': ['S2', 'S3', 'S4'], 'S2': ['S1'], 'S3': ['S1'], 'S4': ['S1']}
S1 => S2 : 0.3333
S1 => S3 : 0.3333
S1 => S4 : 0.3333
S2 => S1 : 1.0000
S3 => S1 : 1.0000
S4 => S1 : 1.0000
```


S13
enter word "...".
click and filter button
0.3333
S3
Filter with 'and filter'
S31
click quick filter button
1.0000
S1
Menu for finding nodes
S14
enter word "...".
click or filter button
0.3333
1.0000
1.0000
0.3333
S21
click quick filter button
S12
enter word "...".
click quick filter button
S41
click quick filter button
S4
Filter with 'or filter'
S2
Filter with quick filter

Function 5

FSM



Log File

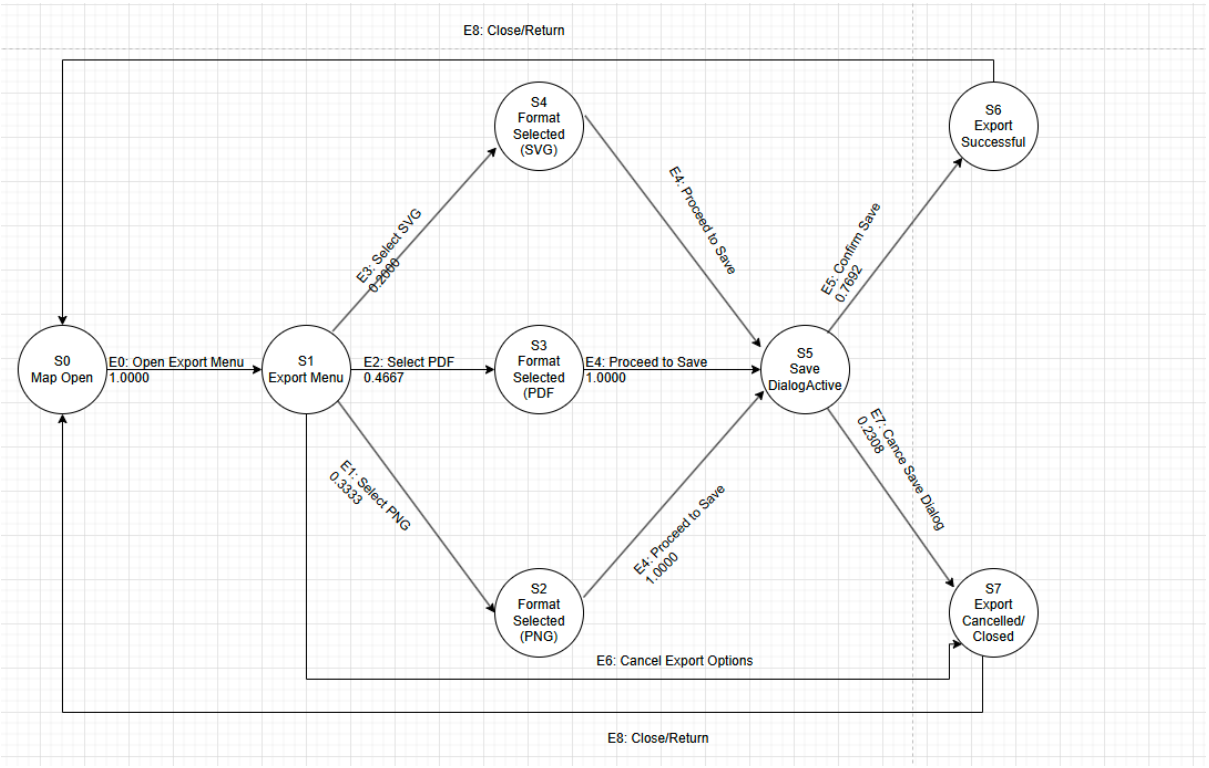
```
From_State,To_State,Count,Probability
S0,S1,15,1.0000
S1,S2,5,0.3333
S1,S3,7,0.4667
S1,S4,3,0.2000
S2,S5,5,1.0000
S3,S5,7,1.0000
S5,S6,10,0.7692
S5,S0,3,0.2308
```

Probability Calculation Code

```python
# analyze_export_fsm.py
from collections import defaultdict

log_file = 'export_fsm.log'

transition_counts = defaultdict(lambda: defaultdict(int))
state_counts = defaultdict(int)

with open(log_file, encoding='utf-8') as f:
    for line in f:
        parts = line.strip().split(',')
        if len(parts) < 5: continue
        from_state, to_state = parts[2], parts[4]
        transition_counts[from_state][to_state] += 1
        state_counts[from_state] += 1

print("From_State,To_State,Count,Probability")
for from_state, to_states in transition_counts.items():
    total = state_counts[from_state]
    for to_state, count in to_states.items():
        prob = count / total if total > 0 else 0
        print(f"{from_state},{to_state},{count},{prob:.4f}")
```

The Markov Chain

```
From_State,To_State,Count,Probability
S0,S1,15,1.0000
S1,S2,5,0.3333
S1,S3,7,0.4667
S1,S4,3,0.2000
S2,S5,5,1.0000
S3,S5,7,1.0000
S5,S6,10,0.7692
S5,S0,3,0.2308
```


E8: Close/Return
S0
Map Open
E0: Open Export Menu
1.0000
S1
Export Menu
E3: Select SVG
0.2000
E2: Select PDF
0.4667
E1: Select PNG
0.3333
S4
Format
Selected
(SVG)
S3
Format
Selected
(PDF
S2
Format
Selected
(PNG)
E4: Proceed to Save
1.0000
E4: Proceed to Save
1.0000
E4: Proceed to Save
1.0000
S5
Save
DialogActive
E5: Confirm Save
0.7692
E7: Cance Save Dialog
0.2308
S6
Export
Successful
S7
Export
Cancelled/
Closed
E6: Cancel Export Options
E8: Close/Return