NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

CS4211 - FORMAL METHODS FOR SOFTWARE ENGINEERING

FINAL REPORT

MODELLING AND VERIFICATION SINGAPORE EXPO USING PAT

Group 10:

Nguyen Tan Sy Nguyen - A0099429L
Nguyen Huu Thanh - A0112069M
Nguyen An Ha - A0113038U
Nguyen Quoc Binh - A0137763X

NOVEMBER 2015

TABLE OF CONTENTS

I. INTRODUCTION

Singapore Expo is the largest meetings, incentives, conferencing, exhibition venues in Singapore and is one of the largest and most completely equipped centers in the whole of Asia.

- There are ten large multipurpose halls, each being 10,000 square meters in size, giving a total floor space of over 100,000 square meters.
- All available space are interconnected and on ground level and column free.
- Ten conference halls are also available, ranging in size from 89 to 844 square meters, suitable for 150 to 1,000 people.
- In addition, there are meeting rooms which can seat 15 to 125 members.

Singapore Expo

1. Expo event modeling

With multiple events being held at the same time, there is an inevitable necessity for visitors to plan their schedules efficiently in such a way that they can visit at many events as possible in the minimum amount of time. On the other side, event organizers also need to book the venues for their events to balance out the hiring cost and the desired capacity.

Expo event modeling offers an automated solution to these problems by modeling all the events and involved processes systematically, verifying the effectiveness of a certain setup and proposing an optimal configuration for each particularly considered circumstance.

2. PAT (Process Analysis Toolkit)

Process Analysis Toolkit (PAT) is a self-contained framework for to support composing, simulating and reasoning of concurrent, real-time systems and other possible domains. It comes with user friendly interfaces, featured model editor and animated simulator.

Most importantly, PAT implements various model checking techniques catering for different properties such as deadlock-freeness, divergence-freeness, reachability, LTL properties with fairness assumptions, refinement checking and probabilistic model checking. To achieve good performance, advanced optimization techniques are implemented in PAT, e.g. partial order reduction, symmetry reduction, process counter abstraction, parallel model checking. So far, PAT has 3502+ registered users from 932+ organizations in 87 countries and regions.

II. PROBLEM DESCRIPTION

1. Description

Event management is the application of project management to the creation and development of large scale events such as festivals, conferences, ceremonies or conventions. It involves studying the brand, identifying the target audience, devising the event concept, planning the logistics and coordinating the technical aspects before actually launching the event.

Within the context of Singapore Expo, the modeling and verification system is an automated method to simulate the event management process and figure out how to organize it in the most effective way. Since the system is limited to no human intervention, it is guaranteed to be less error-prone and more reliable to use in real life.

2. In scope

- Visitor properties
    - Available time
    - Event preference
- Event properties
    - Venue
    - Time
    - Capacity
    - Prerequisite
- Lobby
    - Capacity

3. Out scope

- Problem linked with natural disaster or terror
- Delay or cancel of Expo due to organizer decision
- Personal interest of customers
- Place outside halls

III. SYSTEM MODELING AND PROPERTIES

Expo organizer need efficient solution to model and verifies:

- Resource
- Shops
- Customers and visitors
- Safety and security constraints.
- Concurrent events

In the scope of this project, our group model three typical types of events:

Venue Event:

- Representing the relationship between Venues and Events along multiple days. Each event can be held in one venue for a range of days. Similarly, one venue can only organize one event in one day.
- Each event has prerequisite events that need to be held at least one day before the event. A valid arrangement is when all events can be organized without prerequisite conflict in a given time period.

Visitor Event:

- Representing the relationship between Visitors and Events in a specific day. Each event can satisfy a limit number of visitor at a particular time based on the capacity of that event. Similarly, each visitor can visit one event at a time.

    - If a visitor is visiting an event, he is definitely at lobby which also has a restraining space. A visitor is defined happy if he can visit all events in a limited time, so the system has the responsibility to check if that requirement is reached.

Visitor Event with Probability:

    - Representing the case the one visitor cannot visit all of events because of the limit in time. However, his goal is to spend as much time he has on visiting events in Expo.
    - The probability will show the preference of a visitor for different events. In that situation, the system will calculate the minimum, maximum place he can visit and the minimum, maximum preference.

INPUT

- **Venue-Event:**
    - NUM_EVENTS: number of events.
    - NUM_VENUES: number of venues.
    - EVENT_DUR: the duration of an event.
    - MAX_TIME: the maximum days that events should finish in.
    - EventPrerequisite[NUM_EVENTS][NUM_EVENTS]:
        - If EventPrerequisite[i][j] equal to 1, event i needs to be held before event j.
        - If EventPrerequisite[i][j] equal to 0, event i doesn't need to be held before event j.

- **Visitor-Event:**
    - NUM_EVENTS: number of events in that day.
    - NUM_VISITORS: number of visitors in that day.
    - LOBBY_CAP: lobby capacity.
    - VIS_DUR: the time a visitor will spend in one event.
    - MAX_TIME: the duration of all events. every visitor need to finish visiting before that time.
    - eventAvailable[NUM_EVENTS]:
        - eventAvailable[i] denotes the maximum number of visitors can be served.

- **Probability model:**
    - NUM_EVENTS: number of events in that day.
    - EVENT[NUM_EVENTS]: the list of events that visitor can possibly visit in that day.
    - TIME_SPENT[NUM_EVENTS]: the time estimated to spend on each event.
    - PROBABILITY[NUM_EVENTS]: the probabilities of the visitor choose to visit the corresponding event.
    - TIME_LIMIT: the free time that a visitor has in 1 day to visit events.
    -

IV. EXPERIMENTS

1. Find the minimum amount of time to organize events where some events have dependencies with others

Some events may depend on others. We have a 2-dimensioned array denoted the dependencies:

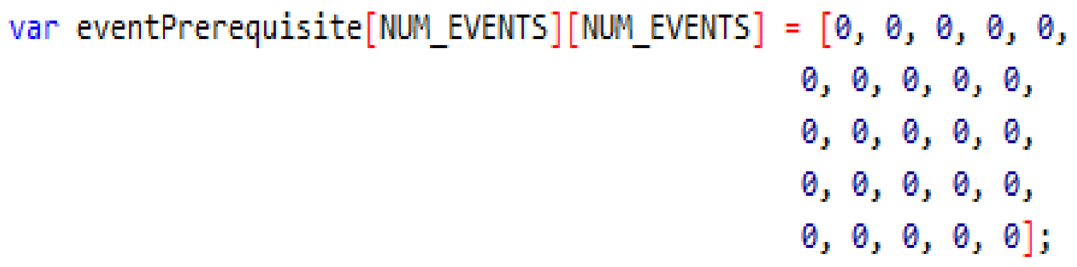
eventPrerequisite(x, y) = 1 if event y can only be organized after event x is organized

This also checks if there is a cycle of dependencies which will make organizing all events impossible.

General setting: there are 5 events, 2 venues, event duration is 1, and maximum time is 5.

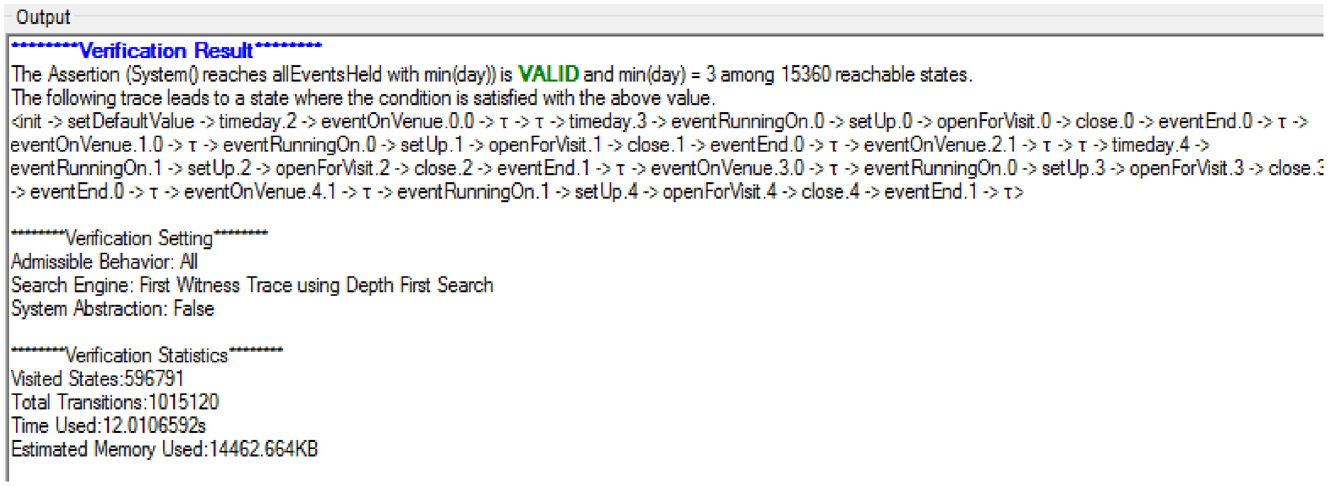
- **Setting A**

No dependencies among all events



```
var eventPrerequisite[NUM_EVENTS][NUM_EVENTS] = [0, 0, 0, 0, 0,
                                                 0, 0, 0, 0, 0,
                                                 0, 0, 0, 0, 0,
                                                 0, 0, 0, 0, 0,
                                                 0, 0, 0, 0, 0];
```

Result: minimum amount of time is 3, one possible configuration is (0, 1) (2, 3) (4)



```
Output
*********Verification Result*********
The Assertion (System() reaches allEventsHeld with min(day)) is VALID and min(day) = 3 among 15360 reachable states.
The following trace leads to a state where the condition is satisfied with the above value.
<init -> setDefaultValue -> timeday.2 -> eventOnVenue.0.0 -> τ -> τ -> timeday.3 -> eventRunningOn.0 -> setUp.0 -> openForVisit.0 -> close.0 -> eventEnd.0 -> τ ->
eventOnVenue.1.0 -> τ -> eventRunningOn.0 -> setUp.1 -> openForVisit.1 -> close.1 -> eventEnd.0 -> τ -> eventOnVenue.2.1 -> τ -> τ -> timeday.4 ->
eventRunningOn.1 -> setUp.2 -> openForVisit.2 -> close.2 -> eventEnd.1 -> τ -> eventOnVenue.3.0 -> τ -> eventRunningOn.0 -> setUp.3 -> openForVisit.3 -> close.
-> eventEnd.0 -> τ -> eventOnVenue.4.1 -> τ -> eventRunningOn.1 -> setUp.4 -> openForVisit.4 -> close.4 -> eventEnd.1 -> τ>

*********Verification Setting*********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

*********Verification Statistics*********
Visited States:596791
Total Transitions:1015120
Time Used:12.0106592s
Estimated Memory Used:14462.664KB
```

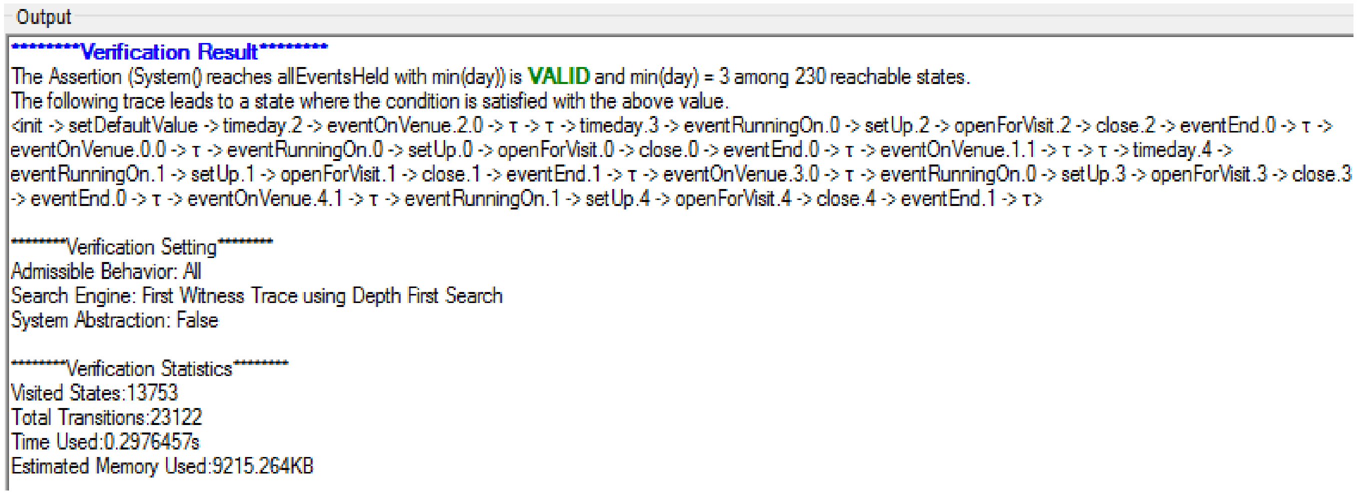
- **Setting B**

Event 2 needs to be organized before event 0 and 1.

Event 0 and 1 both need to be organized before event 3 and 4.

```
var eventPrerequisite[NUM_EVENTS][NUM_EVENTS] = [0, 0, 0, 1, 1,
                                                0, 0, 0, 1, 1,
                                                1, 1, 0, 0, 0,
                                                0, 0, 0, 0, 0,
                                                0, 0, 0, 0, 0];
```

Result: minimum amount of time is 3, one possible configuration is (2) (0 1) (3 4)



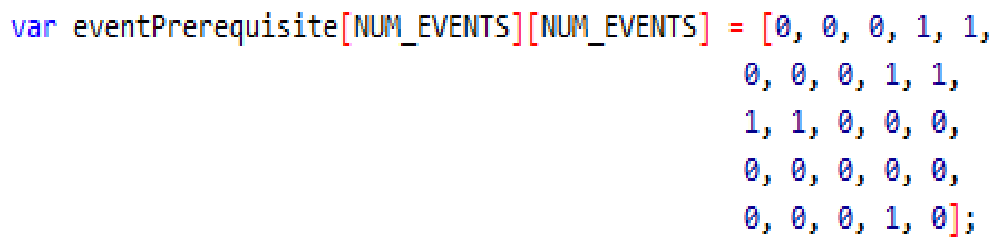
```
Output
*********Verification Result*********
The Assertion (System() reaches allEventsHeld with min(day)) is VALID and min(day) = 3 among 230 reachable states.
The following trace leads to a state where the condition is satisfied with the above value.
<init -> setDefaultValue -> timeday.2 -> eventOnVenue.2.0 -> τ -> τ -> timeday.3 -> eventRunningOn.0 -> setUp.2 -> openForVisit.2 -> close.2 -> eventEnd.0 -> τ ->
eventOnVenue.0.0 -> τ -> eventRunningOn.0 -> setUp.0 -> openForVisit.0 -> close.0 -> eventEnd.0 -> τ -> eventOnVenue.1.1 -> τ -> τ -> timeday.4 ->
eventRunningOn.1 -> setUp.1 -> openForVisit.1 -> close.1 -> eventEnd.1 -> τ -> eventOnVenue.3.0 -> τ -> eventRunningOn.0 -> setUp.3 -> openForVisit.3 -> close.3
-> eventEnd.0 -> τ -> eventOnVenue.4.1 -> τ -> eventRunningOn.1 -> setUp.4 -> openForVisit.4 -> close.4 -> eventEnd.1 -> τ>

*********Verification Setting*********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

*********Verification Statistics*********
Visited States:13753
Total Transitions:23122
Time Used:0.2976457s
Estimated Memory Used:9215.264KB
```

- **Setting C**

Event 2 needs to be organized before event 0 and 1.

Event 0 and 1 both need to be organized before event 3 and 4.

Event 4 needs to be organized before event 3.



```
var eventPrerequisite[NUM_EVENTS][NUM_EVENTS] = [0, 0, 0, 1, 1,
                                                 0, 0, 0, 1, 1,
                                                 1, 1, 0, 0, 0,
                                                 0, 0, 0, 0, 0,
                                                 0, 0, 0, 1, 0];
```

Result: minimum amount of time is 4, one possible configuration is (2) (0 1) (4) (3)

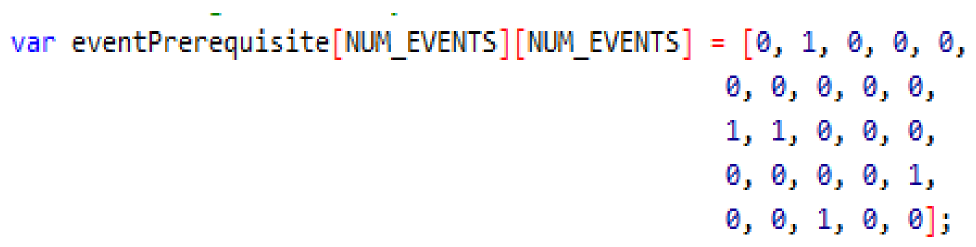
- **Setting D**

Event 0 needs to be organized before event 1

Event 2 needs to be organized before event 0 and 1

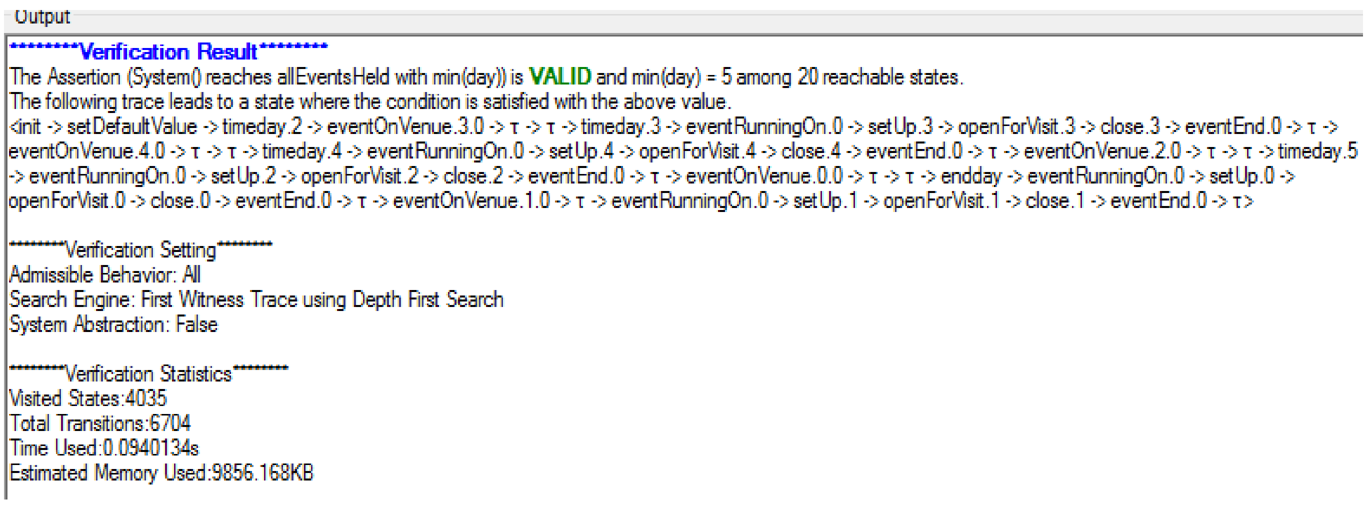
Event 3 needs to be organized before event 4

Event 4 needs to be organized before event 2



```
var eventPrerequisite[NUM_EVENTS][NUM_EVENTS] = [0, 1, 0, 0, 0,
                                                 0, 0, 0, 0, 0,
                                                 1, 1, 0, 0, 0,
                                                 0, 0, 0, 0, 1,
                                                 0, 0, 1, 0, 0];
```

Result: minimum amount of time is 5, one possible configuration is (3) (4) (2) (0) (1)



```
Output
*********Verification Result*********
The Assertion (System() reaches allEventsHeld with min(day)) is VALID and min(day) = 5 among 20 reachable states.
The following trace leads to a state where the condition is satisfied with the above value.
<init -> setDefaultValue -> timeday.2 -> eventOnVenue.3.0 -> τ -> τ -> timeday.3 -> eventRunningOn.0 -> setUp.3 -> openForVisit.3 -> close.3 -> eventEnd.0 -> τ ->
eventOnVenue.4.0 -> τ -> τ -> timeday.4 -> eventRunningOn.0 -> setUp.4 -> openForVisit.4 -> close.4 -> eventEnd.0 -> τ -> eventOnVenue.2.0 -> τ -> τ -> timeday.5
-> eventRunningOn.0 -> setUp.2 -> openForVisit.2 -> close.2 -> eventEnd.0 -> τ -> eventOnVenue.0.0 -> τ -> τ -> endday -> eventRunningOn.0 -> setUp.0 ->
openForVisit.0 -> close.0 -> eventEnd.0 -> τ -> eventOnVenue.1.0 -> τ -> eventRunningOn.0 -> setUp.1 -> openForVisit.1 -> close.1 -> eventEnd.0 -> τ>

*********Verification Setting*********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

*********Verification Statistics*********
Visited States:4035
Total Transitions:6704
Time Used:0.0940134s
Estimated Memory Used:9856.168KB
```

2. Find the minimum amount of time for all visitors to visit all events given a time limit

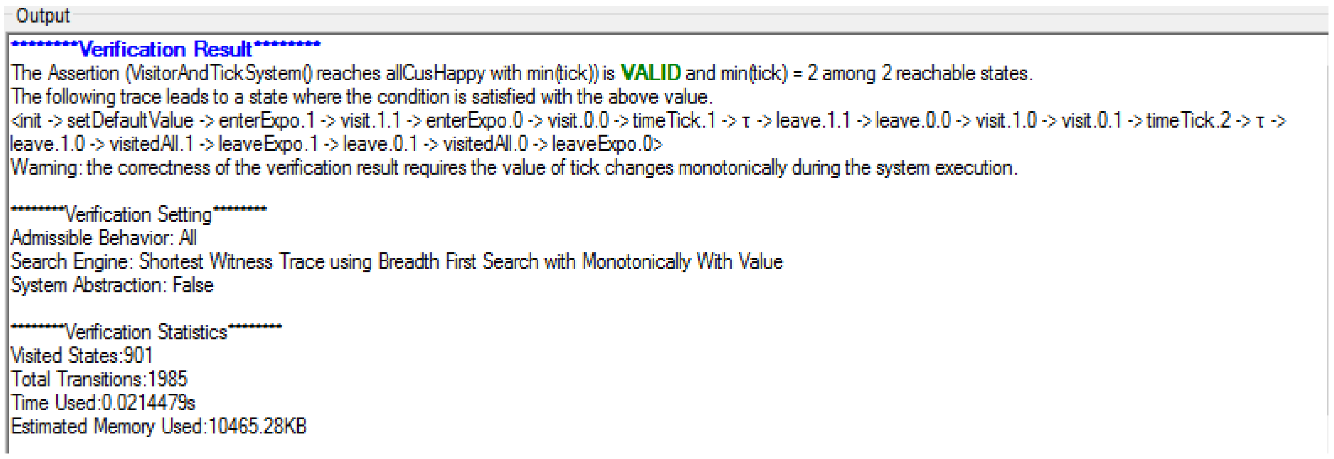
General setting: there are 2 events, the lobby capacity is 20 and the visiting duration is 1.

- **Setting A**

There are 2 visitors and time limit is 2.

```
#define NUM_EVENTS 2;
#define NUM_VISITORS 2;
#define LOBBY_CAP 20;
#define VIS_DUR 1;
#define MAX_TIME 2;
```

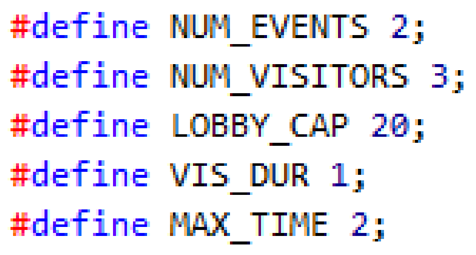
Result: It’s possible for all customers to visit all events within the time limit and minimum amount of time is 2.



```
Output
********Verification Result********
The Assertion (VisitorAndTickSystem() reaches allCusHappy with min(tick)) is VALID and min(tick) = 2 among 2 reachable states.
The following trace leads to a state where the condition is satisfied with the above value.
<init -> setDefaultValue -> enterExpo.1 -> visit.1.1 -> enterExpo.0 -> visit.0.0 -> timeTick.1 -> τ -> leave.1.1 -> leave.0.0 -> visit.1.0 -> visit.0.1 -> timeTick.2 -> τ ->
leave.1.0 -> visitedAll.1 -> leaveExpo.1 -> leave.0.1 -> visitedAll.0 -> leaveExpo.0>
Warning: the correctness of the verification result requires the value of tick changes monotonically during the system execution.

********Verification Setting********
Admissible Behavior: All
Search Engine: Shortest Witness Trace using Breadth First Search with Monotonically With Value
System Abstraction: False

********Verification Statistics********
Visited States:901
Total Transitions:1985
Time Used:0.0214479s
Estimated Memory Used:10465.28KB
```

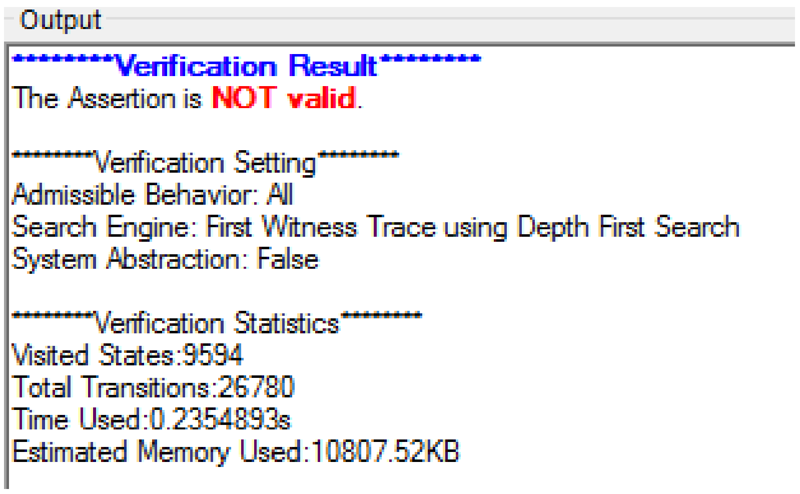
- **Setting B**

There are 3 visitors and time limit is 2.



```
#define NUM_EVENTS 2;
#define NUM_VISITORS 3;
#define LOBBY_CAP 20;
#define VIS_DUR 1;
#define MAX_TIME 2;
```

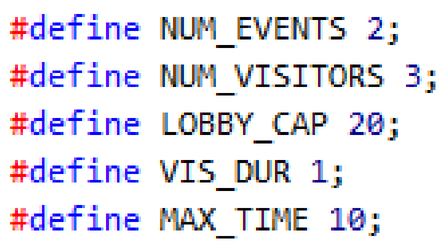
Result: It's impossible for all visitors to visit all events within the time limit.



```
Output
********Verification Result********
The Assertion is NOT valid.

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:9594
Total Transitions:26780
Time Used:0.2354893s
Estimated Memory Used:10807.52KB
```

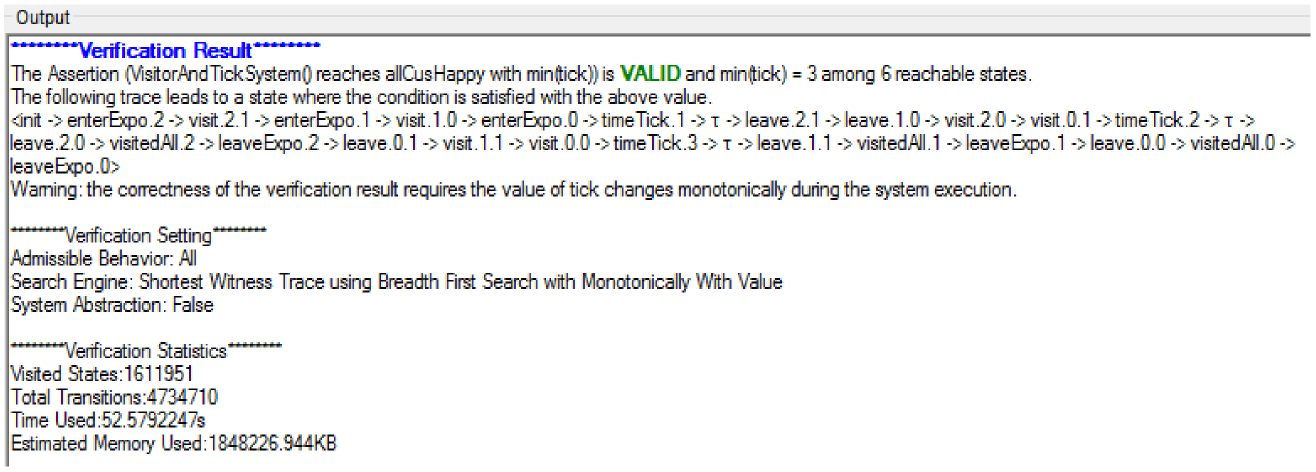
- **Setting C**

There are 3 visitors and time limit is 10.



```
#define NUM_EVENTS 2;
#define NUM_VISITORS 3;
#define LOBBY_CAP 20;
#define VIS_DUR 1;
#define MAX_TIME 10;
```

Results: It’s possible for all visitors to visit all events within the time limit and the minimum amount of time is 3.



```
Output
*********Verification Result*********
The Assertion (VisitorAndTickSystem() reaches allCusHappy with min(tick)) is VALID and min(tick) = 3 among 6 reachable states.
The following trace leads to a state where the condition is satisfied with the above value.
<init -> enterExpo.2 -> visit.2.1 -> enterExpo.1 -> visit.1.0 -> enterExpo.0 -> timeTick.1 -> τ -> leave.2.1 -> leave.1.0 -> visit.2.0 -> visit.0.1 -> timeTick.2 -> τ ->
leave.2.0 -> visitedAll.2 -> leaveExpo.2 -> leave.0.1 -> visit.1.1 -> visit.0.0 -> timeTick.3 -> τ -> leave.1.1 -> visitedAll.1 -> leaveExpo.1 -> leave.0.0 -> visitedAll.0 ->
leaveExpo.0>
Warning: the correctness of the verification result requires the value of tick changes monotonically during the system execution.

*********Verification Setting*********
Admissible Behavior: All
Search Engine: Shortest Witness Trace using Breadth First Search with Monotonically With Value
System Abstraction: False

*********Verification Statistics*********
Visited States:1611951
Total Transitions:4734710
Time Used:52.5792247s
Estimated Memory Used:1848226.944KB
```

3. Find the maximum and minimum number of events a visitor can visit given a time limit and event probabilities

For each event, we know the amount of time required to visit it and the probability a visitor will choose to visit it.

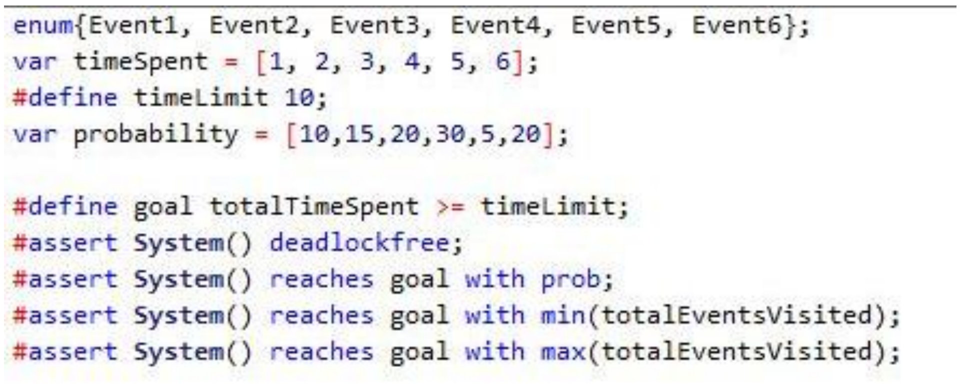
Setting A: There are 6 events:

- Event 1 requires 1 unit of time and has probability of 10%
- Event 2 requires 2 unit of time and has probability of 15%
- Event 3 requires 3 unit of time and has probability of 20%

- Event 4 requires 4 unit of time and has probability of 30%
- Event 5 requires 5 unit of time and has probability of 5%
- Event 6 requires 6 unit of time and has probability of 20%

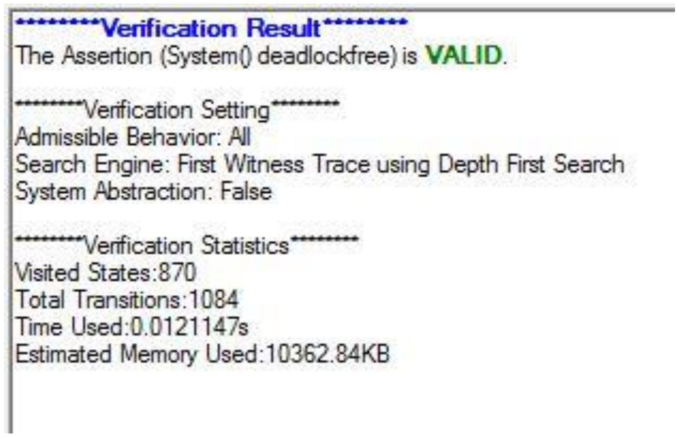
The time limit is 10.

We assume the case that if the visitor exceeds his time limit while visiting an event, then he will proceed to finish visiting that event before leaving the Expo.



```
enum{Event1, Event2, Event3, Event4, Event5, Event6};
var timeSpent = [1, 2, 3, 4, 5, 6];
#define timeLimit 10;
var probability = [10,15,20,30,5,20];

#define goal totalTimeSpent >= timeLimit;
#assert System() deadlockfree;
#assert System() reaches goal with prob;
#assert System() reaches goal with min(totalEventsVisited);
#assert System() reaches goal with max(totalEventsVisited);
```

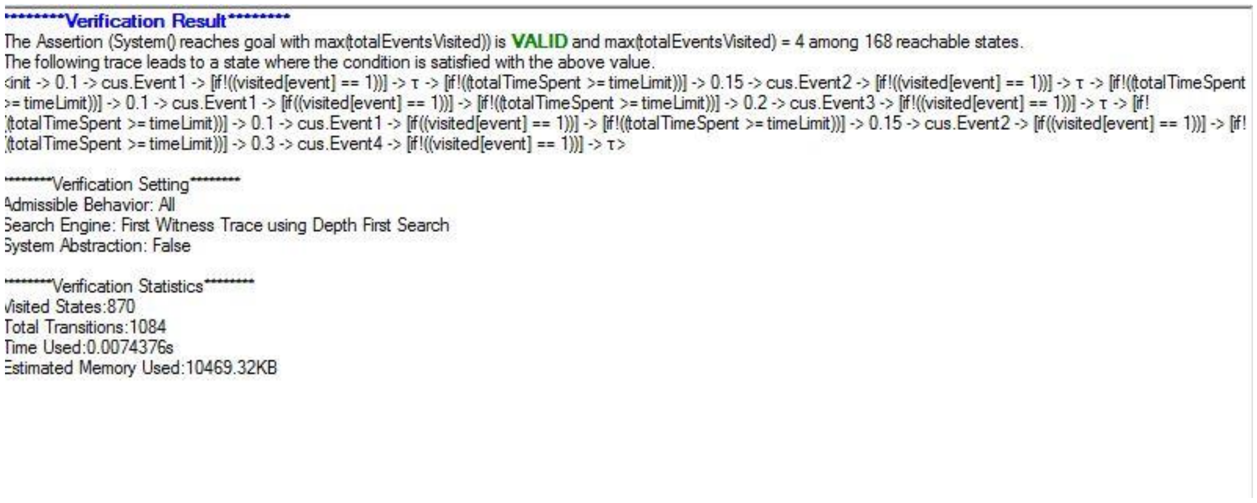
- Assert deadlockfree



```
********Verification Result********
The Assertion (System() deadlockfree) is VALID.

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:870
Total Transitions:1084
Time Used:0.0121147s
Estimated Memory Used:10362.84KB
```

- Maximum number of events visited is 4

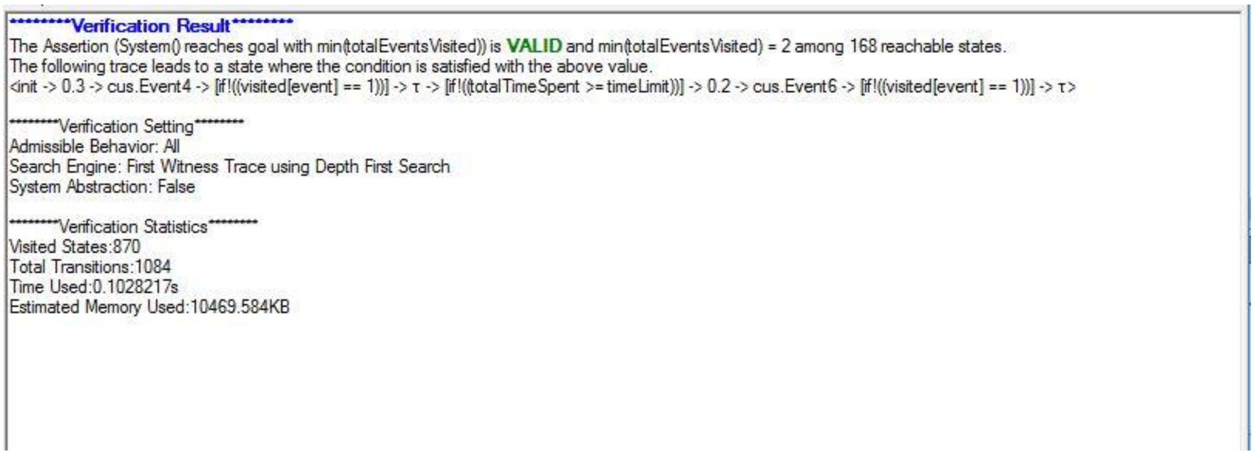
In this case, the events visited are event1, event2, event3, event4 with total time spent is 10.



```
********Verification Result********
The Assertion (System() reaches goal with max(totalEventsVisited)) is VALID and max(totalEventsVisited) = 4 among 168 reachable states.
The following trace leads to a state where the condition is satisfied with the above value.
<init -> 0.1 -> cus.Event1 -> [if!((visited[event] == 1))] -> τ -> [if!((totalTimeSpent >= timeLimit))] -> 0.15 -> cus.Event2 -> [if!((visited[event] == 1))] -> τ -> [if!((totalTimeSpent
>= timeLimit))] -> 0.1 -> cus.Event1 -> [if((visited[event] == 1))] -> [if!((totalTimeSpent >= timeLimit))] -> 0.2 -> cus.Event3 -> [if!((visited[event] == 1))] -> τ -> [if!
((totalTimeSpent >= timeLimit))] -> 0.1 -> cus.Event1 -> [if((visited[event] == 1))] -> [if!((totalTimeSpent >= timeLimit))] -> 0.15 -> cus.Event2 -> [if((visited[event] == 1))] -> [if!
((totalTimeSpent >= timeLimit))] -> 0.3 -> cus.Event4 -> [if!((visited[event] == 1))] -> τ>

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:870
Total Transitions:1084
Time Used:0.0074376s
Estimated Memory Used:10469.32KB
```

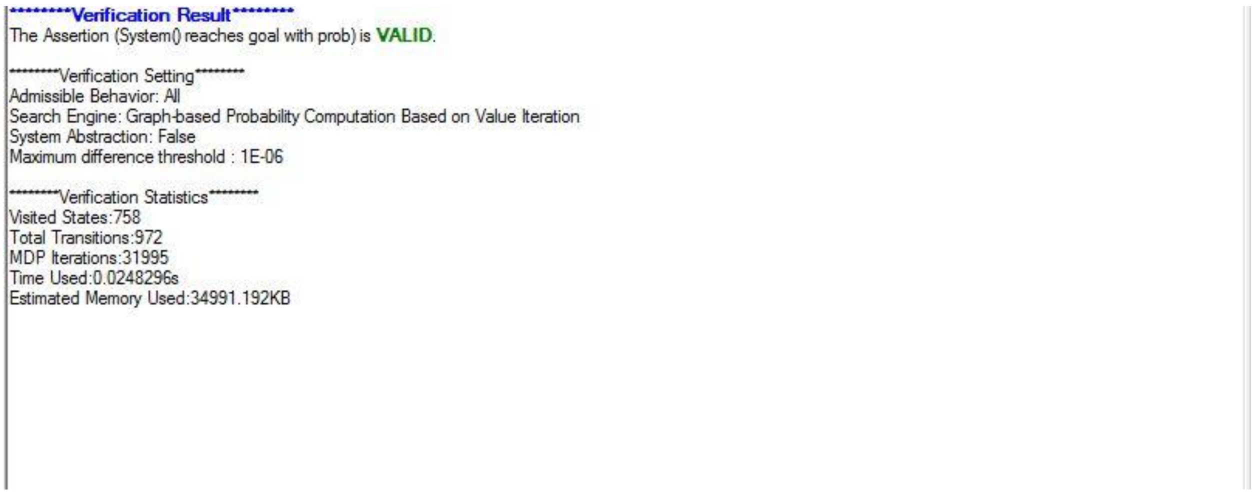
- Minimum number of events visited is 2

  In this case, the events visited are event4, event6 with total time spent is 10.



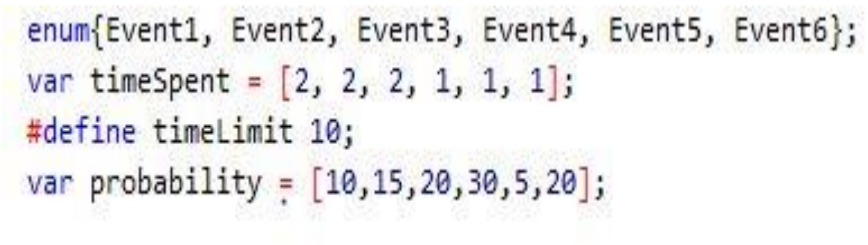
```
********Verification Result********
The Assertion (System() reaches goal with min(totalEventsVisited)) is VALID and min(totalEventsVisited) = 2 among 168 reachable states.
The following trace leads to a state where the condition is satisfied with the above value.
<init -> 0.3 -> cus.Event4 -> [if!((visited[event] == 1))] -> τ -> [if!((totalTimeSpent >= timeLimit))] -> 0.2 -> cus.Event6 -> [if!((visited[event] == 1))] -> τ>

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:870
Total Transitions:1084
Time Used:0.1028217s
Estimated Memory Used:10469.584KB
```

- Assert that there is a case the total visiting time exceeds the time limit



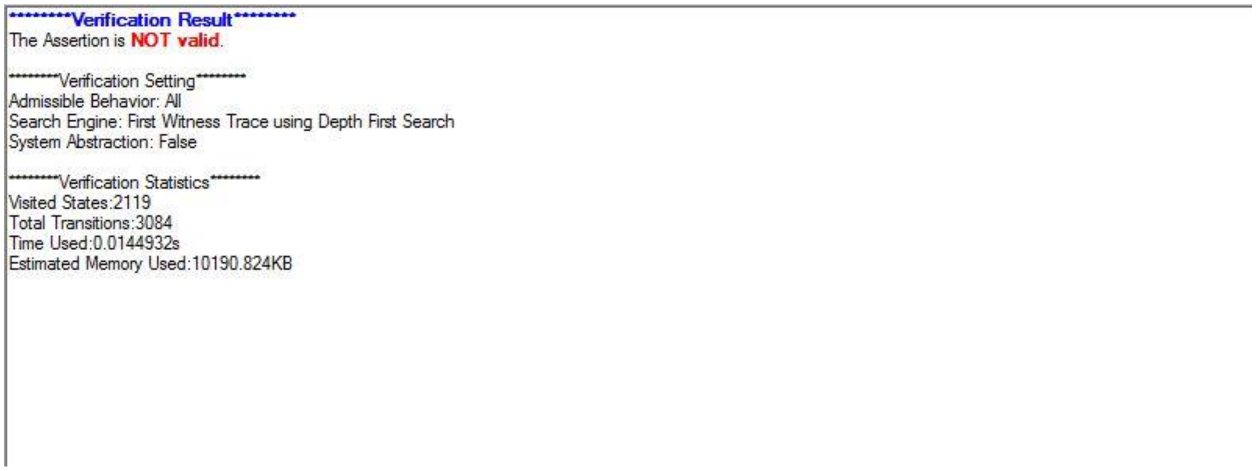
```
********Verification Result********
The Assertion (System() reaches goal with prob) is VALID.

********Verification Setting********
Admissible Behavior: All
Search Engine: Graph-based Probability Computation Based on Value Iteration
System Abstraction: False
Maximum difference threshold : 1E-06

********Verification Statistics********
Visited States:758
Total Transitions:972
MDP Iterations:31995
Time Used:0.0248296s
Estimated Memory Used:34991.192KB
```

Setting B: We keep the same probability set of events, time limit of visitor and change the set of estimated time spent on each event as following:



```
enum{Event1, Event2, Event3, Event4, Event5, Event6};
var timeSpent = [2, 2, 2, 1, 1, 1];
#define timeLimit 10;
var probability = [10,15,20,30,5,20];
```

- In this case, the visitor can never achieve his goal (spends all of his time limit on visiting events) due to the total time spent of all events is less than the time limit given.



```
*********Verification Result*********
The Assertion is NOT valid.

*********Verification Setting*********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

*********Verification Statistics*********
Visited States:2119
Total Transitions:3084
Time Used:0.0144932s
Estimated Memory Used:10190.824KB
```

Hence, both the assertions for minimum events visited and maximum events visited are not valid.

V. FEEDBACK AND SUGGESTION FOR PAT

PAT Version using: 3.5.1 (Build 21590)

1. Bugs report:

- **Words suggestion**: this functionality tends to work as expected when start typing a word, but not able to suggest words after deleting a few characters. Hence we need to delete the whole words and retype to re-enable words suggestion.

2. Suggestions for PAT:

- **Multiple files opener**: allow users to open multiple .csp files at once to ease the process of opening a whole project containing multiple files.
- **Debug mode and File Explorer**: can be included to help users with debugging and managing files.
- **Verification Result Highlight:** beside highlighting result (Valid or Not Valid), we can also highlight the **events** in the trace to see the result clearer.

3. Feedback:

- **GUI**: user-friendly, easy to understand and master the functionalities.
- **Built-in examples:** play important part in helping users get familiar with CSP language and how to verify a model using PAT.

VI. CONCLUSIONS

In conclusions, Expo as a main location for holding important meetings, conference and exhibitions in Singapore needs an automated solution to plan for events and venues such that each visitor has an amazing experience coming to Expo, and Expo Event Modeling offers the solution to this particular problem. Moreover, we are also provided with PAT, which is extremely useful in implementing models checking.

Our Expo Event Modeling is combined of three key models: Event-Venue, Visitor-Event and Probability Model for Visitor. Event-Venue model can make sure whether an Event can only be held in a Venue given the prerequisite of such Event. Therefore, visitors are always able to visit the event they want and all of the prerequisite events. Visitor-Event model can make sure that even though events are limited in capacities, all visitors will be able to visit the events that they demand. Probability Model is the solution for how many events a visitor are able to visit in his time limit given the probabilities of him going to each event.

Therefore, with these three models combined, Expo Event Modeling is promised to ease the process of booking and planning events for both events host and visitors.

VII. REFERENCES

1. Jun Sun, Yang Liu, Jin Song Dong and Jun Pang. PAT: Towards Flexible Verification under Fairness. The 21th International Conference on Computer Aided Verification (CAV 2009), pages 709-714, Grenoble, France, June, 2009.
2. Jun Sun, Yang Liu, Jin Song Dong. PAT Introduction.
   Source: http://pat.sce.ntu.edu.sg/