CS5232

FORMAL SPECIFICATION AND DESIGN TECHNIQUES

Project: Safety alarm system for car

Name: Truong Viet Trung

Matric. Number: A0099324X

Table of Contents

1. Abstract

Automatic alarm system is used more and more in all products around, from sound alarm when fire occurs to fridge door is opened longer than expected. As such, there is a need for automatic alarm system for vehicles (in this case cars) who taking part in traffic. An alarm could help user notice when safety standards are being violated by himself, and take actions accordingly. This project is not implemented in this module, however, modelling using PAT really bring significantly valuable information before the actual implementation of the system. The models will show how car's headlight changes corresponding to external light automatically, internal light changes based on various reason such as door open/close or ambient light, door alarm when the doors are not fully closed when engine starts, collision alarm when there is possibility that the car will hit the obstacle on its path.

2. Project Description

The objective of this project is to emulate some functionalities of a car. Different tasks must be created and managed simultaneously to collect the various information.

Here are the details for the models:

2.1 Headlight

The headlight needs to be able to respond to external light source automatically. Here is how it should respond:

- There is a threshold value for external light so that we can safety say that it is dark when brightness level is below that threshold.
- When external light is bright, headlight should be OFF.
- When external light is dark, headlight should be ON.

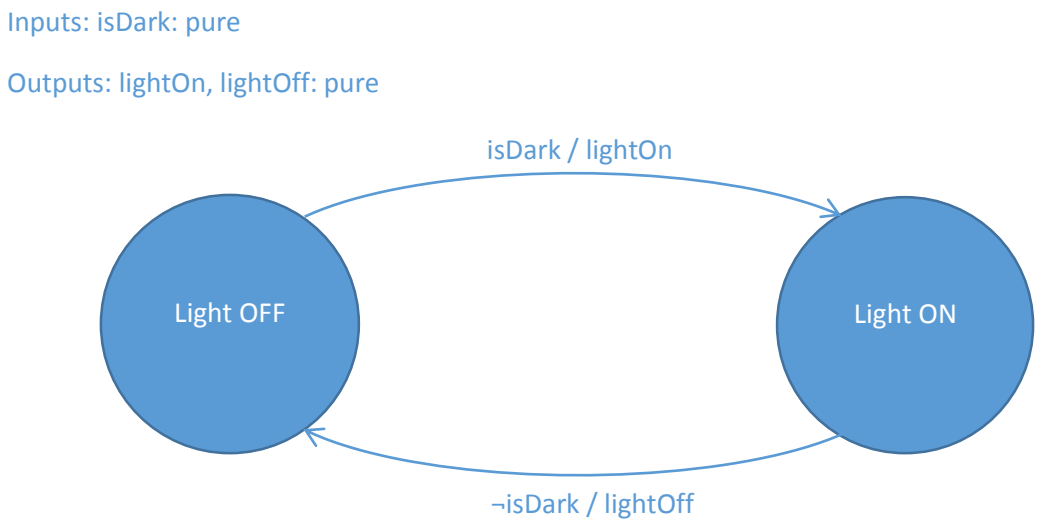


Figure 1: State Machine Diagrams for headlight

2.2 Internal light

Internal light is the light inside a car. This light should be on/off based on various condition. Here is the details of internal light:

- The interior lights are turned on as soon as the door is opened if the external ambient light is below the threshold (i.e. exterior is dark).
- If the door remains open for 20 sec, the lights are turned OFF.
- If the door is closed during the 20 sec interval then the lights are ON only for 10 sec more and then slowly dimmed OFF.
- When the door is closed, lights are turned ON.
- If the door has been closed for 10 sec, the lights are slowly dimmed OFF.
- If the engine starts while the lights are ON, then the lights are immediately switched OFF irrespective of the state of the door.

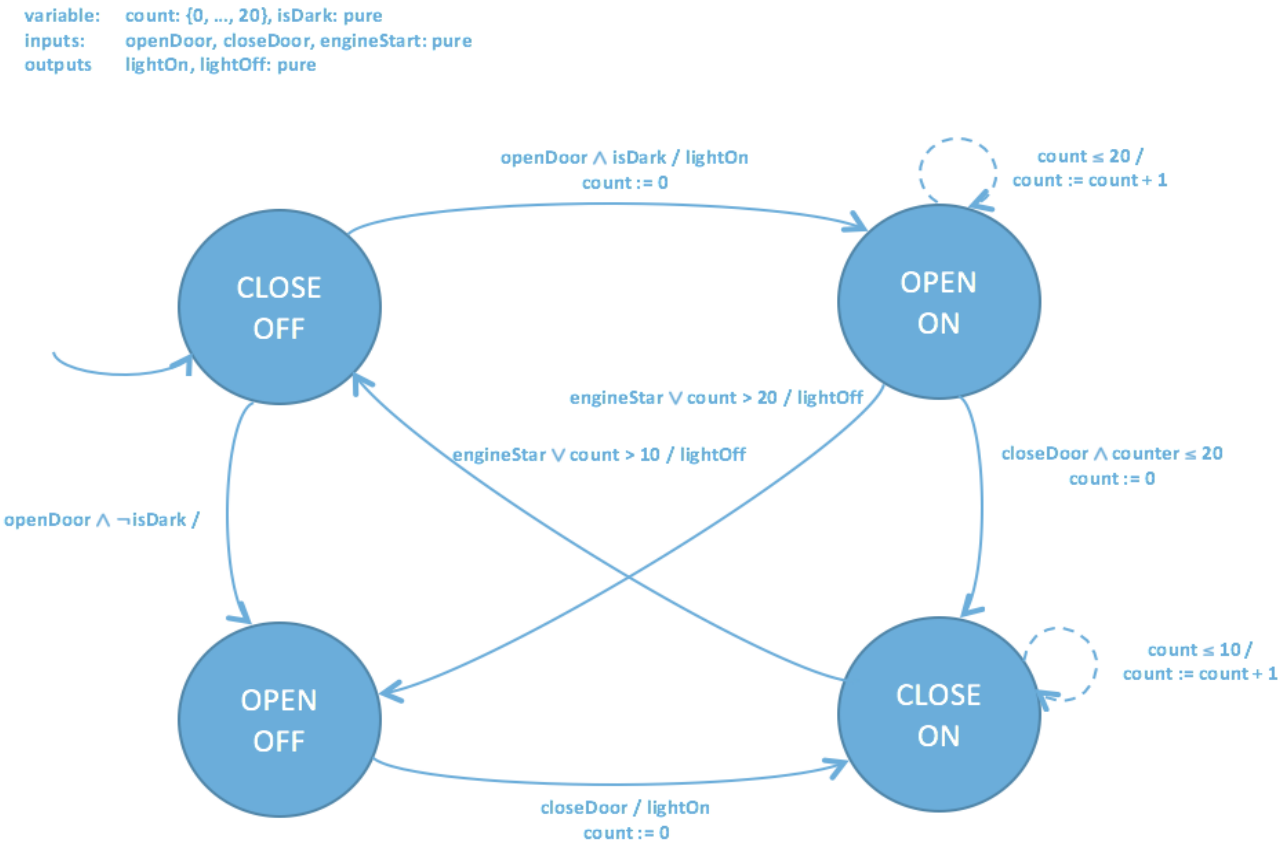


Figure 2: State Machine Diagrams for internal light

2.3 Unsafe Door Alarm

This alarm is corresponding to the state where the door is opened but the car engine is running. This is because when a car is running it is dangerous to have door opened. The alarm can be deactivated by either

- Closing the opened door, or
- Stopping the engine

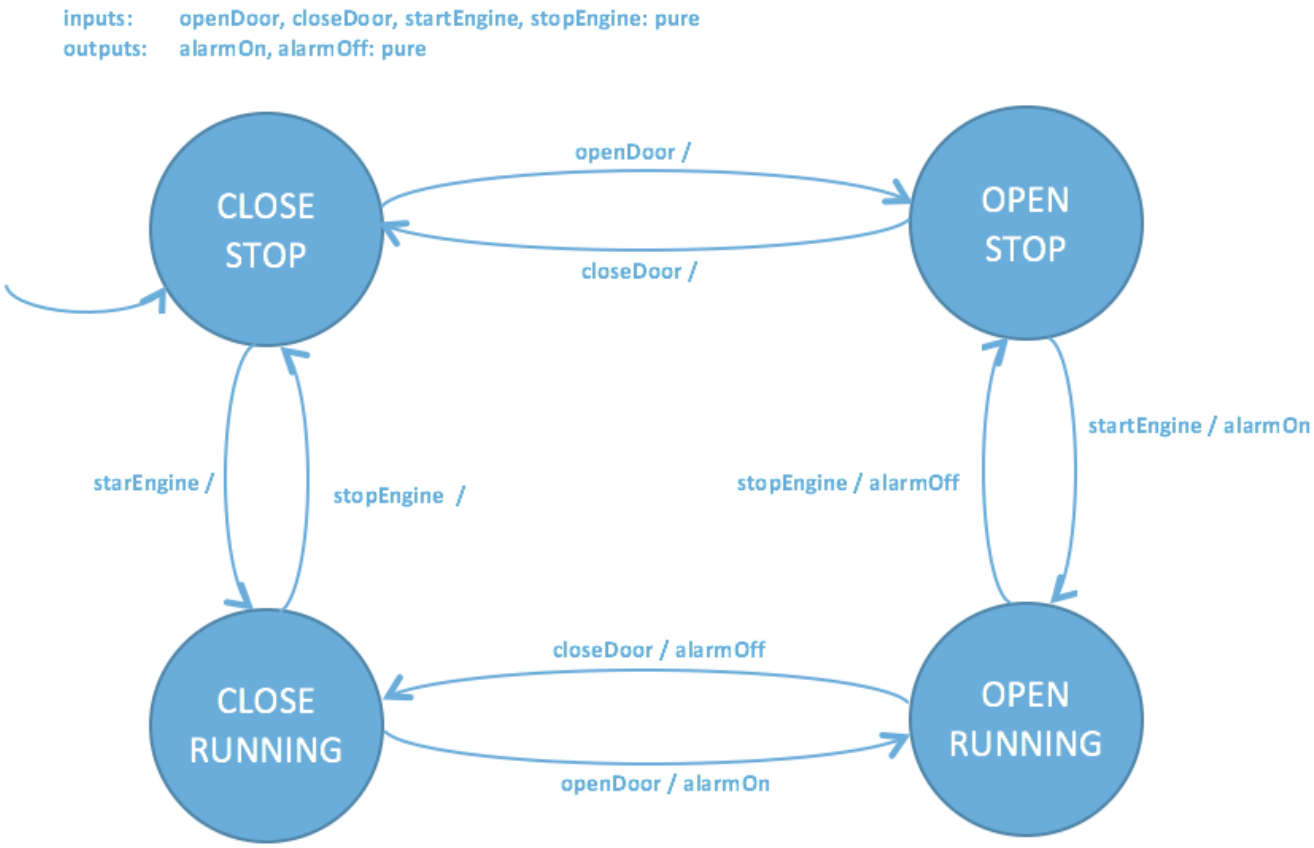


Figure 3: State Machine Diagrams for unsafe door alarm

2.4 Obstacle ahead alarm

This alarm is used to alert the driver whenever there is a chance that the driver will hit the obstacle in front of him. When this alarm is on, the car will automatically apply the brake and try to stop itself before hitting the target.

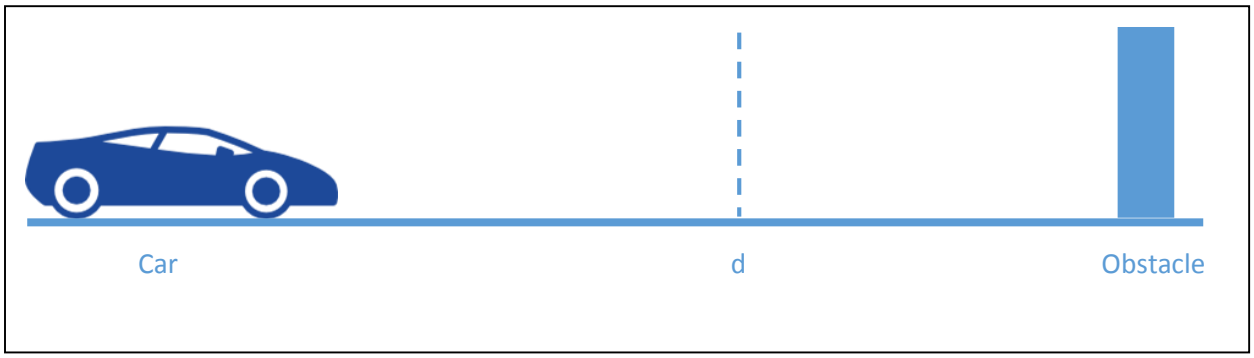


Figure 4: Car and obstacle diagram

For the collision alarm to activate, the car must be in a position where it could collide with the obstacle ahead.

In *Figure 4*, when the car reach *d*, the system does the following:

- Get the current speed as well as deceleration by braking of the car
- Calculate the distance away for the obstacle (possibly using supersonic/infrared sensor)
- Calculate the distance travelled by the car if it starts applying the brake
- If the latter distance is even or greater than the former one, alarm activates

When the alarm activates, it means that the car could hit the obstacle even if it starts to brake. Therefore it is important to apply the brake in order to avoid collision with the obstacle.

2.5 User stops the engine manually

This is usually when the car speed reach zero. However, sometimes the user can stop the car's engine even when it is running (speed is not zero). In this case all the car system will be off and the car would still be moving due to inertia. When the engine stops but car speed is not zero, due to the weight of the car as well as friction with the road, the car will slow down until it finally stops (it will take longer time to slow down compared to applying brake). Thus, the car could hit the obstacle when the user stop the engine and the collision alarm is on. To fix this, the system

disable engine to be off so that the car will be able to apply the brake by itself to stop the car quicker and to avoid colliding with the obstacle.

3. System Modelling

3.1 Model design

There are two parts to the whole system: the environment and the car.

At the moment the environment only consists of daylight. The level of brightness of daylight directly affects the headlight of the car.

The car consists of two models: the door controller which affects internal light as well as unsafe door alarm, and the engine controller which affects the speed of the car and collision prediction alarm.

A timer is also used to measure the time in second (for door) and it is used for scheduling when to turn off the internal light (according to 2.2).

3.2 Assumptions

3.2.1 Energy

The engine is running on unlimited energy so that there is no sudden stop of the engine due to lack of energy. The only way for engine to stop is through user's interaction.

3.2.2 Environment

Apart from daylight we assume other environment conditions remain constant or does not have significant change to the car (road friction, weather, etc.)

3.2.3 Automation and user's interaction

We assume that the user will take minimal control of the car. In the model, the user only do open/close door and start/stop the engine.

The following tasks are automated: headlight on/off, internal light on/off, alarm activate/deactivate, and brake when collision possibility alarm is on.

3.2.4 Car

The car has constant acceleration (thus constant speed).

There is a max speed so that the speed of the car will never exceed that speed. The speed of the car cannot become negative either.

3.2.5 Light

In this model light (headlight, interior light and daylight) is a Boolean value (on/off, bright/dark). This is used because it is easier to simulate the states using the program.

Internal light will not be on when engine is running.

3.3 Processes

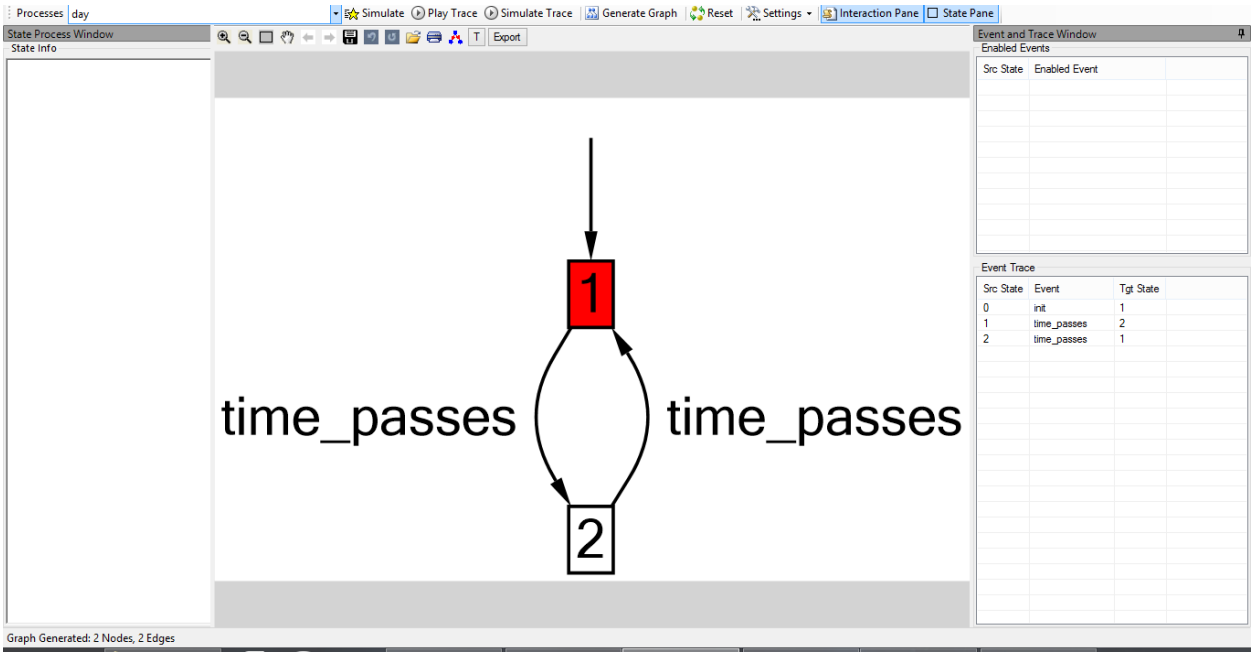
`system`: two parallel processes `environment` and `car`

`environment`: one process `day`

`day: 2 events`

- When it is bright it becomes dark
- When it is dark it becomes bright

  Below is the simulation graph for day

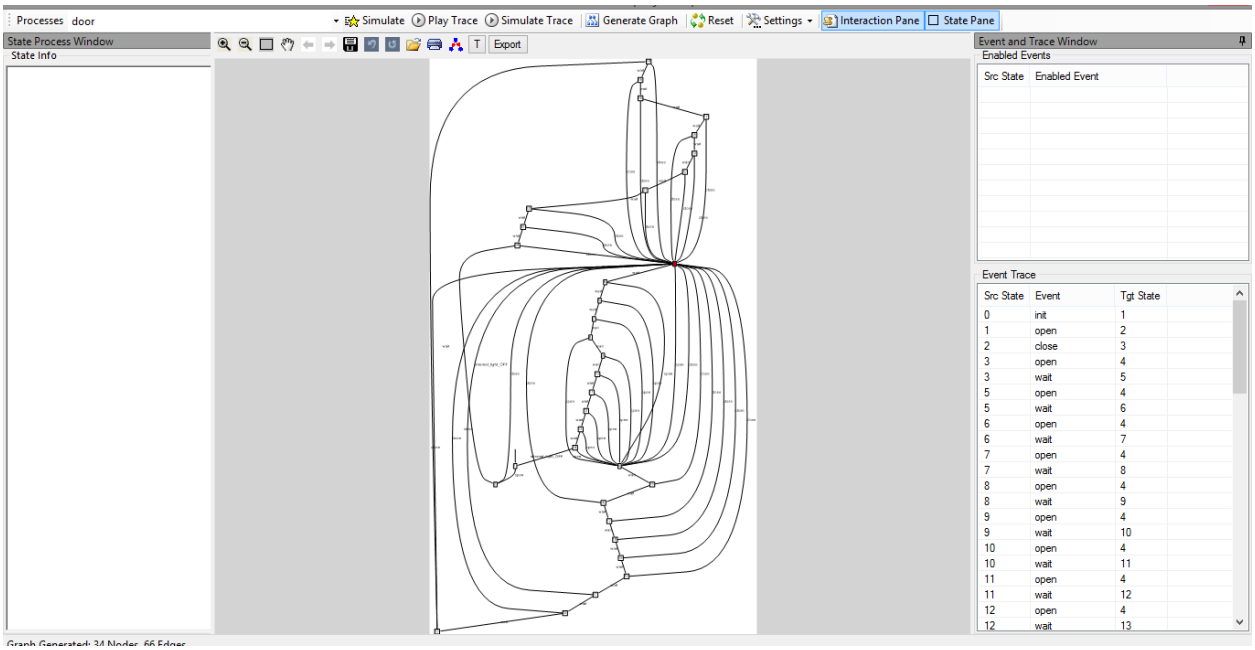


`car`: two parallel processes `door` and `engine`

`door`: 7 events

- Open door when external light is bright
- Open door when external light is dark
- Wait for internal light to be on for 20 seconds
- Turn internal light off after 20 seconds
- Close door
- Wait for internal light to be on for 10 seconds
- Turn internal light off after 10 seconds

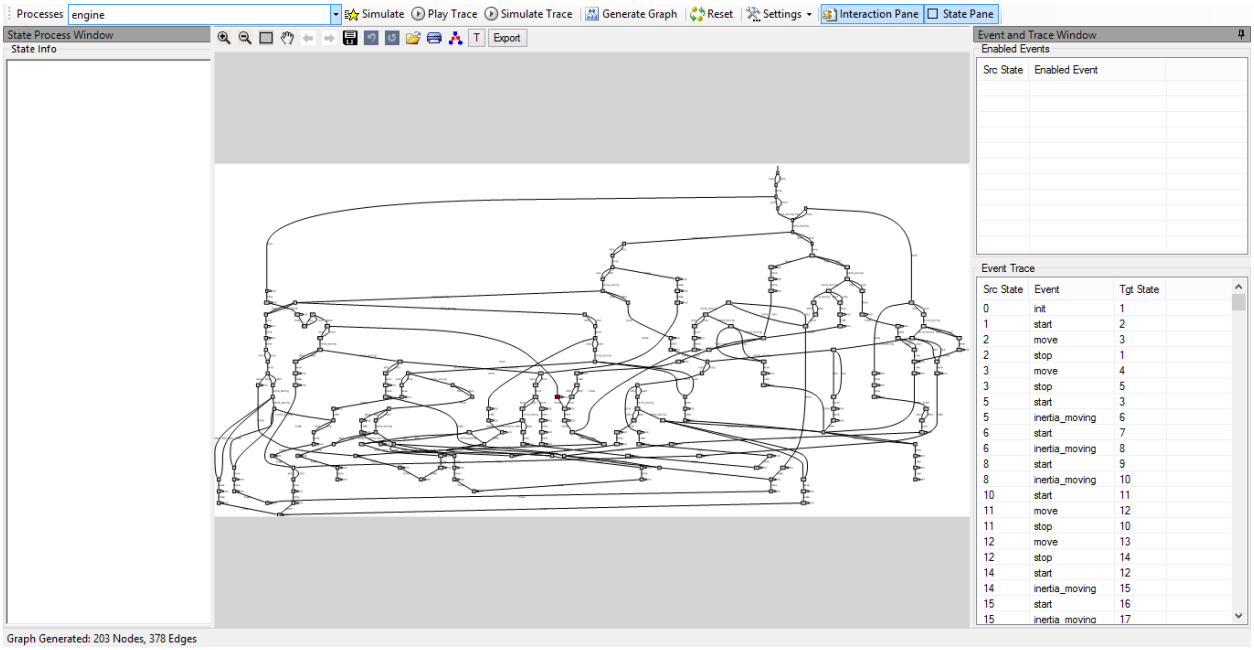
  Below is the simulation graph by PAT for door



`engine`: 5 events

- Start engine
- Move the car
- Apply brake
- Let the car move by inertia
- Stop engine

  Below is the simulation graph by PAT for engine



3.4 Events

3.4.1 Door

Events	Conditions	Description
open	Door is closed and external light is bright	Door is opened
open	Door is closed and external light is dark	Door is opened, internal light is on if engine is not running
wait	Door is opened, internal light is on for less than 20 sec	Idle
internal_light_OFF	Door is opened, internal light is on for 20 sec or more	Turn off internal light

close	Door is opened	Door is closed, any unsafe door alarm is deactivated, if engine is not running internal light is on
wait	Door is closed, internal light is on for less than 10 sec	Idle
internal_light_OFF	Door is closed, internal light is on for 10 sec or more	Turn off internal light

3.4.2 Engine

Events	Conditions	Description
start	Engine is off	If the car will hit the obstacle when engine is on and moving, engine continues to be off, otherwise on. If the door is opened, unsafe door alarm activates. Internal light is turned off
move	Engine is on and no alarm is activated (both unsafe door alarm and possible collision alarm)	Calculate speed, distance travelled by the car. Calculate if the car applies the brake now will it be able to stay in a safe distance from obstacle, if not, possible collision alarm is on.

brake	Engine is on, possible collision alarm is on	Apply brake. If the speed reach zero stop the engine and deactivate possible collision alarm. If the car hit the obstacle signal that collision occurs.
inertia_moving	Engine is off (by the user), speed is positive (car is still moving)	Car slows down, calculate distance moved by the car afterward. If it hits the obstacle signal that collision occurs.
stop	Engine is on	If possible collision alarm is not on, turn off the engine. This is to disable the user ability to turn off the engine manually and possibly let it move with inertia to hit the obstacle. Turn off any unsafe door alarm.

3.4.3 Daylight

Events	Conditions	Description
time_passes	Daylight is bright	Simulate a day. After bright will be dark. Car headlight will react to ambient light being dark and turn on.

time_passes	Daylight is dark	Simulate a day. After dark will be bright again. Car headlight will react to ambient light being bright and turn off.

3.5 Constants

Constants	Value	Description
off	0	The state of something is off (light, alarm, engine)
on	1	The state of something is on (light, alarm, engine)
closed	0	The state of door is closed
opened	1	The state of door is opened
dark	0	The state of dark ambient light
bright	1	The state of bright ambient light
obstaclePosition	INTEGER; subject to change	The position of the obstacle from the car's start position. This value could be changed for different experiments
max_speed	INTEGER; subject to change	The possible maximum speed of a car. This value could be changed for different experiments
min_speed	0	The minimum speed of a car. Zero because it is not moving.

acceleration	INTEGER; subject to change	The acceleration applies to the car. This value could be changed for different experiments
brake_deceleration	INTEGER; subject to change	The brake deceleration applies when brake is used. This value could be changed for different experiments.
normal_deceleration	INTEGER; subject to change	The normal deceleration due to friction, this applies when car engine is off (no braking happens). This value need to be smaller than brake_deceleration. It could be changed for different experiments

3.6 Variables

Variables	Description	Default value
engineState	The state of the car engine	off
doorState	The state of the car door	closed
daylight	The state of daylight	bright
unsafeDoorAlarm	The state of unsafe door alarm. This alarm is on when door opens and engine is on	off
obstacleAheadAlarm	The state of possible collision alarm. This alarm is on when applying brake could still	off

	make the car reach a too near distance from the obstacle	
headlightState	The state of the headlight	off
internalLightState	The state of the internal light	off
timeCounterOpen	The time counter for internal light when door is opened	0 (INTEGER)
timeCounterClose	The time counter for internal light when door is closed	0 (INTEGER)
distance	The distance travelled by the car	0 (INTEGER)
collided	The signal whether the car is collided with the obstacle	false
speed	The speed of the car	min_speed (INTEGER)

4. Investigated Properties

4.1 Deadlock free

```
#assert system deadlockfree;
```

This assertion is to verify that the system will not encounter a deadlock state.

4.2 Divergence free

```
#assert system divergencefree;
```

This assertion is to verify that the system has no process that perform internal transitions forever without engaging any useful events

4.3 Deterministic

```
#assert system deterministic;
```

This assertion is to verify that the system has no process with an event leads to two or more different states.

4.4 Non terminating

```
#assert system nonterminating;
```

The system is nonterminating because there is not only daylight cycle but door and engine cycle as well.

4.5 Car collides with obstacle when alarm is on

```
#define collideWhenObstacleAheadAlarmOn (obstacleAheadAlarm == on && collided == true);

#assert system reaches collideWhenObstacleAheadAlarmOn;
```

The system will verify whether the car will collide with the obstacle when the possible collision is on. If the program is correct this should never happens because the car is forced to brake and stop before hitting the obstacle.

4.6 Car collides with obstacle

```
#define collideWithObstacle (collided == true);

#assert system reaches collideWithObstacle;
```

The system will verify whether the car will collide with the obstacle. Sometimes this happens and some time it is not. This is due to different conditions (constants and variables) that affects the system. Sometimes it is not possible but sometime it is. If it is possible this is due to the user stops then engine when it is running with high speed (before the possible collision alarm is on), and it will move with inertia and eventually hit the obstacle (this is actually very dangerous therefore user might be better not off the engine when car is running at all).

4.7 Internal light on when engine is on

```
#define internalLightOnWhenEngineOn (internalLightState == on && engineState == on);

#assert system reaches internalLightOnWhenEngineOn;
```

The system should not reach it states because whenever engine is on that internal light is off immediately (and the user is not allowed to manually turned on the internal light)

4.8 Moving when alarm is on

```
#define movingWhenAlarmOn (speed > 0 && (obstacleAheadAlarm == on || unsafeDoorAlarm == on));
```

```
#assert system reaches movingWhenAlarmOn;
```

This assertion is valid because there is a chance that the user open the door when the car is moving. (After this the unsafe door alarm will be on but this alarm does not force the engine to stop or the door to close)

4.9 Headlight is on when daylight is bright

```
#define headlightOnWhenBright (daylight == bright && headlightState == on);

#assert system reaches headlightOnWhenBright;
```

This assertion will not valid because the headlight is reacting to the daylight and it should be off when daylight is bright.

4.10 Headlight is off when daylight is dark

```
#define headlightOffWhenDark (daylight == dark && headlightState == off);

#assert system reaches headlightOffWhenDark;
```

Similar to 4.9. The headlight should be on when daylight is dark.

4.11 Car exceeds maximum speed

```
#define exceedMaxSpeed (speed > max_speed);

#assert system reaches exceedMaxSpeed;
```

The car should never be able to exceed the limit

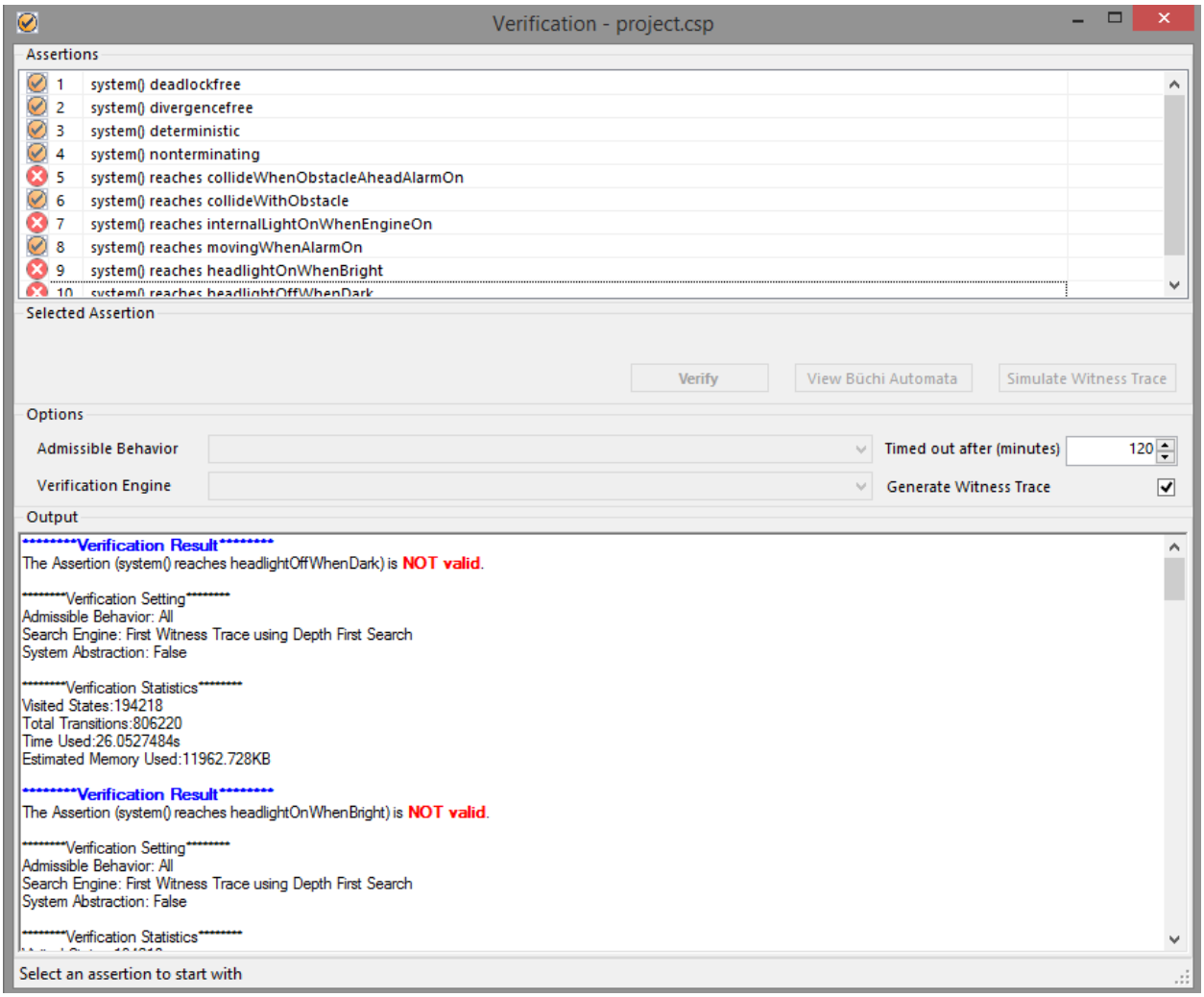
4.12 Car has negative speed

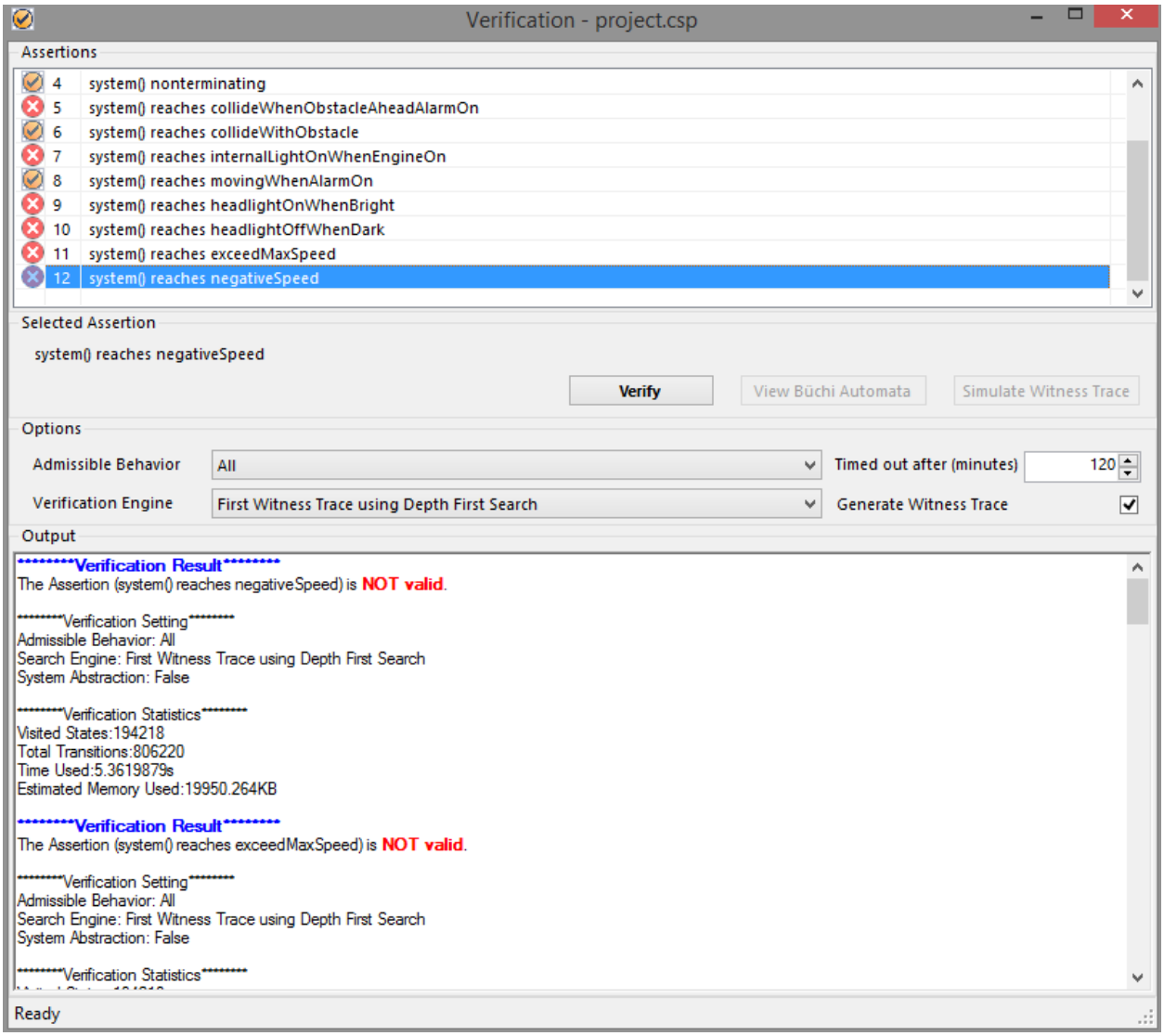
```
#define negativeSpeed (speed < min_speed);

#assert system reaches negativeSpeed;
```

We assume that the car will not move backward

Below are the screenshots of verification results from PAT.


Verification - project.csp
Assertions
1 system() deadlockfree
2 system() divergencefree
3 system() deterministic
4 system() nonterminating
5 system() reaches collideWhenObstacleAheadAlarmOn
6 system() reaches collideWithObstacle
7 system() reaches internalLightOnWhenEngineOn
8 system() reaches movingWhenAlarmOn
9 system() reaches headlightOnWhenBright
Selected Assertion
Verify
View Büchi Automata
Simulate Witness Trace
Options
Admissible Behavior
Timed out after (minutes)
120
Verification Engine
Generate Witness Trace
Output
********Verification Result********
The Assertion (system() reaches headlightOffWhenDark) is NOT valid.
********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False
********Verification Statistics********
Visited States:194218
Total Transitions:806220
Time Used:26.0527484s
Estimated Memory Used:11962.728KB
********Verification Result********
The Assertion (system() reaches headlightOnWhenBright) is NOT valid.
********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False
********Verification Statistics********
Select an assertion to start with


Verification - project.csp
Assertions
4 system() nonterminating
5 system() reaches collideWhenObstacleAheadAlarmOn
6 system() reaches collideWithObstacle
7 system() reaches internalLightOnWhenEngineOn
8 system() reaches movingWhenAlarmOn
9 system() reaches headlightOnWhenBright
10 system() reaches headlightOffWhenDark
11 system() reaches exceedMaxSpeed
12 system() reaches negativeSpeed
Selected Assertion
system() reaches negativeSpeed
Verify
View Büchi Automata
Simulate Witness Trace
Options
Admissible Behavior
All
Timed out after (minutes)
120
Verification Engine
First Witness Trace using Depth First Search
Generate Witness Trace
Output
********Verification Result********
The Assertion (system() reaches negativeSpeed) is NOT valid.
********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False
********Verification Statistics********
Visited States:194218
Total Transitions:806220
Time Used:5.3619879s
Estimated Memory Used:19950.264KB
********Verification Result********
The Assertion (system() reaches exceedMaxSpeed) is NOT valid.
********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False
********Verification Statistics********
Ready

5. Experiments and Observations

A number of different experiments is carried out with a different combination of obstacle position, max speed, acceleration, brake deceleration, and normal deceleration to verify the system.

	Obstacle Position	Max speed	Acceleration	Brake Deceleration	Normal Deceleration
Run 1	1000	200	30	90	10
Run 2	500	200	30	90	10
Run 3	1000	100	30	90	10
Run 4	500	100	30	90	10
Run 5	1000	200	50	90	10
Run 6	500	100	50	90	10
Run 7	1000	200	30	50	10
Run 8	500	100	50	50	10
Run 9	1000	200	30	90	20
Run 10	500	100	50	50	20

After the experiments, in general all the verified results are correct. Only the assertion about car collides with obstacle (4.6) sometimes changes because it is possible in certain experiment that all traces cannot bring the car to collide with obstacle.

6. Limitations, Discussions and Possible Extensions

6.1 Brightness is represented by 0 or 1

It is relative for different people to define darkness. Someone might still want the headlight to be on when its 6 in the morning. Therefore, an improvement could be made so that brightness can be represented as a value, and there is a threshold for different user so that whenever the brightness value is below that threshold it is dark.

6.2 User has limited interaction

No manual internal light control, headlight control, speed control, etc. It is indeed limited for the user in this project. However, this project is created in order to build an automated alarm system that could help the user detect unsafe conduct. Therefore user interaction should be kept at minimal. Later we could expand the project so that the user is given more control over the car and see how user interacts with automated alarm system.

6.3 Infinite energy

In our assumption we state that the car will never runs of energy (it is refilled after each run). A possible extension is to have fuel in the car, and after time the amount of fuel decreases. Another alarm could be added to the system so that it will activate when the fuel is low, and probably automatically stop the engine when the fuel is critical (running on very low fuel makes engine less durable).

6.4 Obstacle is static

Since the obstacle is not moving, a lot of the calculation is on theory and probably not realistic because in real life traffics are moving. However, the system can also easily adapt to moving obstacle because we can apply physics relativity theory. Assume that the obstacle is static, then the car is moving at relative_speed = obstacle_speed – car_speed. This can be used in the system and thus it is extensible for moving obstacle.

6.5 Speed limit

No driver would drive max speed in town. Therefore a possible extension for the project is add a speed limit alarm for the user, and it activates when the speed exceed the recommended limit. Nevertheless, for the modelling using PAT, it is easily done by changing the max_speed value.

6.6 Car only moves forward

At the moment we only consider the car moving toward the obstacle. One extension is that the car speed can be negative which means the car is moving further from the obstacle.

Moreover, we might also improve the project so that it supports the car to move in more directions (left, right, diagonal, etc.)

6.7 Environment conditions is negligible apart from daylight

Other environment conditions could affects the speed of the car as well as the safety of the driver like heavy rain, dampening path, uphill, etc. The project have an environment process added to prepare for such extension on external conditions.

7. Suggestions and Feedbacks

- Debugging messages can be more details so that it is easier to debug. For example, this message appears when we used += or *= (not supported by PAT yet): ‘’{’:invalid symbol:’{’’
- Layout issues (especially simulation windows). When clicking on Interaction Pane and Selection Pane button for a few times, strange layout issues occur
- Suggestion on verification window: probably the assertions pane can be made resizable so that it is more convenient to see all the assertions.

8. Conclusion

In summary, using PAT modeling techniques helps us achieved a general understanding of the system. We have also achieved the goal that enable the alarm to notifies user of unsafe driving conduct, as well as help driver avoid possible collision to certain extent. The system is also functioning correctly after verifying with assertions. In closing, PAT is indeed a good tool that enabled us to model and verify a problem domain in an efficient way.