Traffic Flow

Matteo Zambon

Cyber Physical Systems Exam

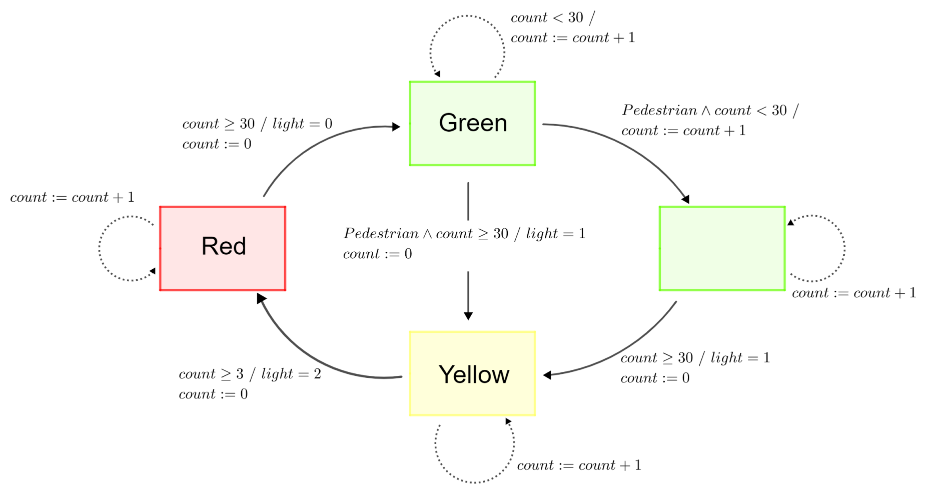
1 Environment Description

The project consists of a representation of a possible real-life situation. The two main objects are:

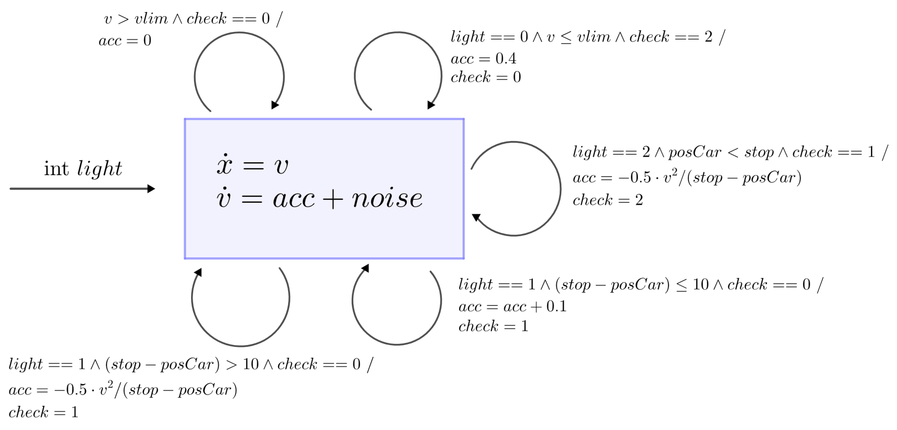
- A traffic light, ideally equipped with a camera that detects the presence of pedestrians waiting to cross the street;
- An autonomous car also equipped with a camera able to discern whether the traffic light is green, yellow or red.

1.1 Design of the CPSs

The traffic light has a discrete internal clock the is reset every time the light change. Yellow and red have a fixed duration, 3 and 30 seconds, while the green stands until the sensor detect a pedestrian, then it sends a new input to the traffic light. The following is the Extended State Machine of the object

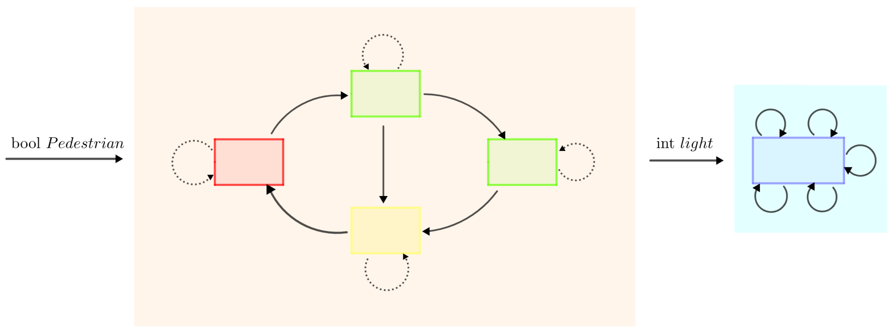


The car follows a continuous external clock, and its action are determined by the input, that is the output of the traffic light, and by the state of the car, meaning position, velocity and acceleration. To add a non-deterministic component, the acceleration has a small random noise to better represent a real-life environment. Position and velocity are affected indirectly by this, but they do not have a personal "noise" component. Now, the car knows (or detects with the sensors) the speed limit on the street, so it stops accelerating after reaching that value. Still, the noise on the acceleration might increase or decrease its velocity. When the camera sees the red light, a signal is sent to the computer, and the car should stop before reaching the traffic light.


$v > vlim \land check == 0$ /
$acc = 0$
$light == 0 \land v \leq vlim \land check == 2$ /
$acc = 0.4$
$check = 0$
int $light$
$\dot{x} = v$
$\dot{v} = acc + noise$
$light == 2 \land posCar < stop \land check == 1$ /
$acc = -0.5 \cdot v^2/(stop - posCar)$
$check = 2$
$light == 1 \land (stop - posCar) \leq 10 \land check == 0$ /
$acc = acc + 0.1$
$check = 1$
$light == 1 \land (stop - posCar) > 10 \land check == 0$ /
$acc = -0.5 \cdot v^2/(stop - posCar)$
$check = 1$

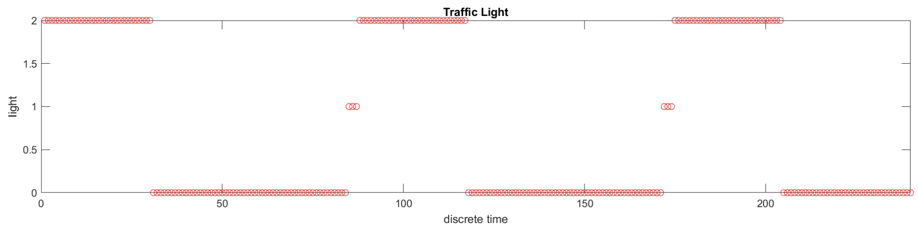
One last internal condition is that, when the speed reach the value 0, the derivative of the velocity is set to 0, regardless of the noise.

We can synthesize in a unique schema as an extended state machine

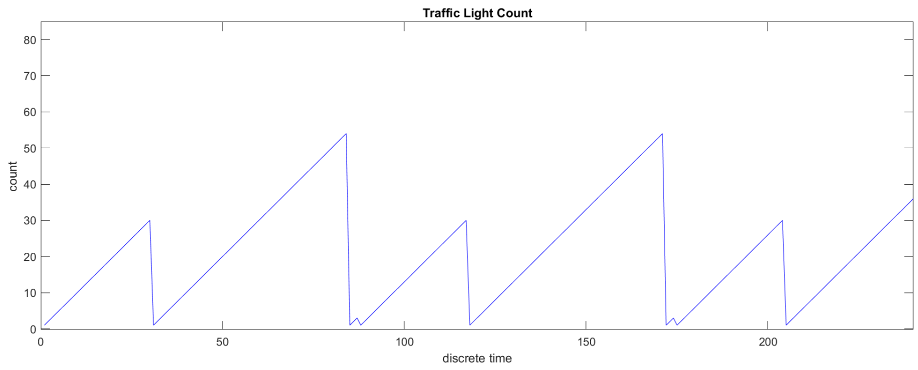


2 Coding and Running

The computation has a time horizon $T = 240$ seconds. The initial speed of the car is equal to the speed limit, and the traffic light is initialized as red. The car then begins to decelerate until it reaches the stop with zero velocity. After the internal clock of the traffic light reaches 30, the light becomes green and the count is reset, the car starts moving. The presence of pedestrians is determined by a function that uses a random number to decide whether the variable is 0 or 1.

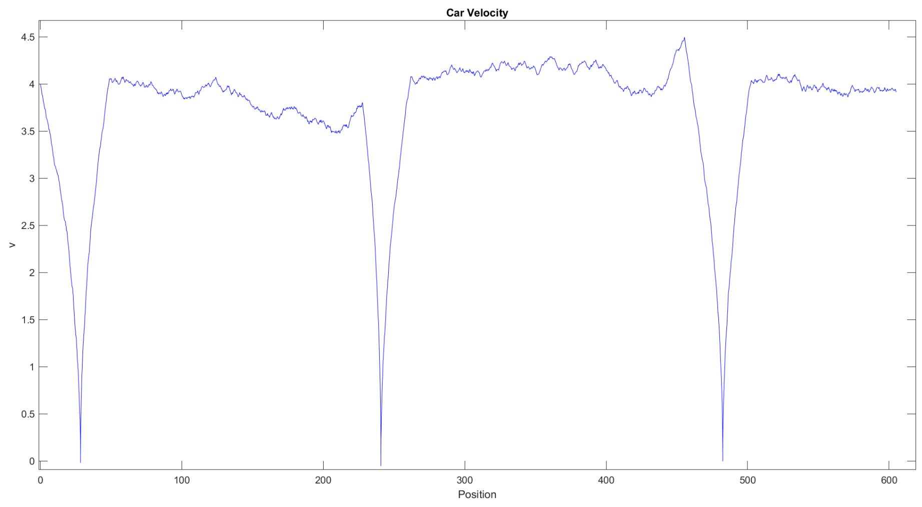


The probability of pedestrians varies based on the interval of time at which we are: the idea was to decrease the probability of $Pedestrian = 1$ in those moments of the day where people are sleeping, while increasing it during peak hours. Since the time horizon is small, we can't see well enough the difference. The duration of the green light is slightly higher sometimes. Increasing the time horizon would produce higher peaks in the traffic light count.



In the velocity plot, we can see clearly the influence of the noise, making the car often exceed the speed limit and possibly getting fined. This can be avoided adjusting the acceleration in a way that it always tends to speed limit, whether it the current speed is higher or lower.

There is an “interesting” behavior right before slowing down, and it is an acceleration. When the light becomes yellow, the car has two choices: if it's far enough, it starts decelerating so that the car stops before the traffic light (hopefully); if it's close enough, it increases its acceleration to pass the traffic light before it turns red. This explains the two peaks.



3 Falsification and Verification

The process of Falsification and Verification has been done using S-Taliro tool on MATLAB. The formulas considered where

$$\phi_1 = \mathbf{G}(v \leq 0 \rightarrow posCar \leq stop)$$
$$\phi_2 = \mathbf{G}(v \leq 4.2 \wedge (light == 1 \rightarrow acc \leq 0.4)) \wedge \mathbf{G}(\mathbf{F}(light == 2))$$

The first formula is simpler, checks that when the car stops because of the red light, it happens before the stop sign. The robustness computed with the function *fw_taliro()* changes a lot between computations: the noise on the acceleration makes the car stop after the traffic light sometimes, and in those cases the robustness has always a negative value; in other cases the formula is satisfied, like in this one, where the robustness value is 0.1257.

The second formula is longer and has 4 predicates. The first predicate in particular is sometimes not satisfied because the speed exceed the limit (for example it happens in the peak between 400 and 500). In this case the robustness is negative and has value -0.4228. The violations are not that bad, it often happens that the speed exceed the limit much more than this particular case, leading the robustness to a value around -2.