**Cache Money** SENG 5801

Mike Galvin HW6

Nathan Glimsdale 12/5/2014

Stephen Stack

Street Light State Machine Verification

***With NuSMV***

## Scope

The scope of this modelling effort is to encapsulate the behavior of a typical traffic intersection with two traffic lights that are responsible for controlling the flow of traffic on each perpendicular street. The intersection has loop sensors embedded in the pavement which detect when cars are waiting at the intersection, buttons that pedestrians can use to send a request to cross and sensors to detect when emergency vehicles are en route on a given street.

## Assumptions/Requirements

The following assumptions about the environment and system can be made:

* The pedestrians will use the same lights as the traffic in the same direction.
* The system will be provided inputs from sensors including pedestrian walk buttons, loop sensors and emergency vehicle sensors.
* The system shall not allow both lights to be green.
* The system shall allow pedestrians and vehicles a reasonable amount of time to cross the street, even if a request from the opposite direction has been received.

## Happy-Path NuSMV Session

-- specification AG (nsLight.currentState = Green -> ewLight.currentState = Red) is true

-- specification AG (nsLight.currentState = Yellow -> ewLight.currentState = Red) is true

-- specification AF (ewLight.currentState = Green -> ((EX ewLight.currentState = Green & EX (EX ewLight.currentState = Green)) & EX (EX (EX ewLight.currentState = Green)))) is true

-- specification AF (nsLight.currentState = Green -> ((EX nsLight.currentState = Green & EX (EX nsLight.currentState = Green)) & EX (EX (EX nsLight.currentState = Green)))) is true

-- specification AG (nsLight.currentState = Red -> AF nsLight.currentState = Green) is true

-- specification AG (nsLight.currentState = Green -> AF nsLight.currentState = Red) is true

-- specification AG (ewLight.currentState = Red -> AF ewLight.currentState = Green) is true

-- specification AG (ewLight.currentState = Green -> AF ewLight.currentState = Red) is true

-- specification AG ((ewPedestrianSignal = TRUE & nsLight.currentState = Green) -> AX nsLight.currentState = Yellow) is true

-- specification AG ((nsPedestrianSignal = TRUE & ewLight.currentState = Green) -> AX ewLight.currentState = Yellow) is true

-- specification AG ((ewPedestrianSignal = TRUE & nsLight.currentState = Green) -> AF nsLight.currentState = Red) is true

-- specification AG ((nsPedestrianSignal = TRUE & ewLight.currentState = Green) -> AF ewLight.currentState = Red) is true

-- specification AG ((ewLight.currentState = Green & nsPedestrianSignal = TRUE) -> AF nsLight.currentState = Green) is true

-- specification AG ((nsLight.currentState = Green & ewPedestrianSignal = TRUE) -> AF ewLight.currentState = Green) is true

-- specification AG ((ewLight.currentState = Green & nsLoopSignal = TRUE) -> AF nsLight.currentState = Green) is true

-- specification AG ((nsLight.currentState = Green & ewLoopSignal = TRUE) -> AF ewLight.currentState = Green) is true

-- specification AG ((ewLight.currentState = Green & nsEmergencyVehicleSignal = TRUE) -> AF nsLight.currentState = Green) is true

-- specification AG ((nsLight.currentState = Green & ewEmergencyVehicleSignal = TRUE) -> AF ewLight.currentState = Green) is true

-- specification !(AG (ewLight.currentState = Green & nsLight.currentState = Green)) is true

-- specification AG ((nsPedestrianSignal = TRUE & ewLight.currentState = Green) -> ((EX ewLight.currentState = Yellow & EX (EX ewLight.currentState = Yellow)) & EX (EX (EX ewLight.currentState = Yellow)))) is true

-- specification AG ((nsLoopSignal = TRUE & ewLight.currentState = Green) -> ((EX ewLight.currentState = Yellow & EX (EX ewLight.currentState = Yellow)) & EX (EX (EX ewLight.currentState = Yellow)))) is true

-- specification AG ((nsEmergencyVehicleSignal = TRUE & ewLight.currentState = Green) -> ((EX ewLight.currentState = Yellow & EX (EX ewLight.currentState = Yellow)) & EX (EX (EX ewLight.currentState = Yellow)))) is true

## Anti-Property 1

**Natural Language:**

For all paths where the north-south light is green the east-west light is also green.

**CTL:**

SPEC AG(nsLight.currentState=Green -> (ewLight.currentState=Green)) ANTIPROPERTY

**Transcript:**

Trace Type: Counterexample

-> State: 1.1 <-

ewPedestrianSignal = FALSE

ewEmergencyVehicleSignal = FALSE

ewLoopSignal = FALSE

nsPedestrianSignal = FALSE

nsEmergencyVehicleSignal = FALSE

nsLoopSignal = FALSE

ewLight.currentState = Green

ewLight.greenTimer = 60

ewLight.yellowTimer = 10

nsLight.currentState = Red

nsLight.greenTimer = 60

nsLight.yellowTimer = 10

controller.activeLight = ew

-> State: 1.2 <-

nsPedestrianSignal = TRUE

ewLight.greenTimer = 59

controller.activeLight = ns

-> State: 1.3 <-

nsPedestrianSignal = FALSE

ewLight.currentState = Yellow

ewLight.greenTimer = 58

-> State: 1.4 <-

ewLight.yellowTimer = 9

-> State: 1.5 <-

ewLight.yellowTimer = 8

-> State: 1.6 <-

ewLight.yellowTimer = 7

-> State: 1.7 <-

ewLight.yellowTimer = 6

-> State: 1.8 <-

ewLight.yellowTimer = 5

-> State: 1.9 <-

ewLight.yellowTimer = 4

-> State: 1.10 <-

ewLight.yellowTimer = 3

-> State: 1.11 <-

ewLight.yellowTimer = 2

-> State: 1.12 <-

ewLight.yellowTimer = 1

-> State: 1.13 <-

ewLight.yellowTimer = 0

-> State: 1.14 <-

ewLight.currentState = Red

ewLight.yellowTimer = 10

-> State: 1.15 <-

nsLight.currentState = Green

## Anti-Property 2

**Natural Language:**

In all cases if a light is green then the next step is yellow.

**CTL:**

**AG (nsLight.currentState = Green -> AX nsLight.currentState = Yellow)**

## Transcript:

Trace Type: Counterexample

-> State: 1.1 <-

ewPedestrianSignal = FALSE

ewEmergencyVehicleSignal = FALSE

ewLoopSignal = FALSE

nsPedestrianSignal = FALSE

nsEmergencyVehicleSignal = FALSE

nsLoopSignal = FALSE

ewLight.currentState = Green

ewLight.greenTimer = 60

ewLight.yellowTimer = 10

nsLight.currentState = Red

nsLight.greenTimer = 60

nsLight.yellowTimer = 10

controller.activeLight = ew

-> State: 1.2 <-

nsPedestrianSignal = TRUE

ewLight.greenTimer = 59

controller.activeLight = ns

-> State: 1.3 <-

nsPedestrianSignal = FALSE

ewLight.currentState = Yellow

ewLight.greenTimer = 58

-> State: 1.4 <-

ewLight.yellowTimer = 9

-> State: 1.5 <-

ewLight.yellowTimer = 8

-> State: 1.6 <-

ewLight.yellowTimer = 7

-> State: 1.7 <-

ewLight.yellowTimer = 6

-> State: 1.8 <-

ewLight.yellowTimer = 5

-> State: 1.9 <-

ewLight.yellowTimer = 4

-> State: 1.10 <-

ewLight.yellowTimer = 3

-> State: 1.11 <-

ewLight.yellowTimer = 2

-> State: 1.12 <-

ewLight.yellowTimer = 1

-> State: 1.13 <-

ewLight.yellowTimer = 0

-> State: 1.14 <-

ewLight.currentState = Red

ewLight.yellowTimer = 10

-> State: 1.15 <-

nsLight.currentState = Green

-> State: 1.16 <-

nsLight.greenTimer = 59

controller.activeLight = ew