ECE 597MB/622 - Spring 2022
Modeling and Verification of Embedded Systems
Lab #2 Timed Automata Verification using UPPAAL

Objectives of this lab:

1) To understand symbolic analysis of timed automata;
2) To gain exposure to modeling and verification of real-time systems using Uppaal;

Sources:

Uppaal: http://www.uppaal.org/

Install version **4.1.19** of UPPAAL. You can find tutorial documents and detailed descriptions of how to use UPPAAL on the website. The 32-bit Windows version can be used on 64-bit Windows.

General lab submission instructions:

1) Lab reports should be typed, with name and ID at the beginning of the Lab report.
2) All screenshots and text should be clearly readable.
3) Your submission **must** include the source files (in .xml format) of your UPPAAL models and queries as indicated in the lab assignment. A single .xml file will contain both the model and query if you are using the correct version of UPPAAL (4.1.19). If you are unsure about this, close the file and re-open it to check that the queries are saved.
4) You can work alone or in groups of two. If you work with a partner, you must complete step 4 of part B. If you work alone, you can skip that step and will still receive the points.

Introduction:

Hybrid systems are systems that combine discrete and continuous dynamics. Timed automata are the simplest non-trivial hybrid systems. Most real-time systems require measuring the passage of time and performing actions at specific times. A device that measures the passage of time, a clock, has a particularly simple dynamics: its state progresses linearly in time. In this lab we will use UPPAAL to check the properties of a real-time system.

UPPAAL is an integrated tool environment for modeling, simulation and verification of real-time systems, developed jointly by Aalborg University in Denmark and Uppsala University in Sweden. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with real-valued clocks, communicating through channels or shared variables. Typical application areas include real-time controllers and communication protocols where timing aspects are critical.

Questions:

Part A (30 points)

Part A is to help you learn to use UPPAAL to design and verify a real-time system. After installing UPPAAL, you can find that there is a demo folder in your UPPAAL installation path. Open the “bridge” file in the demo folder (File->Open System) to find a model of the following system:

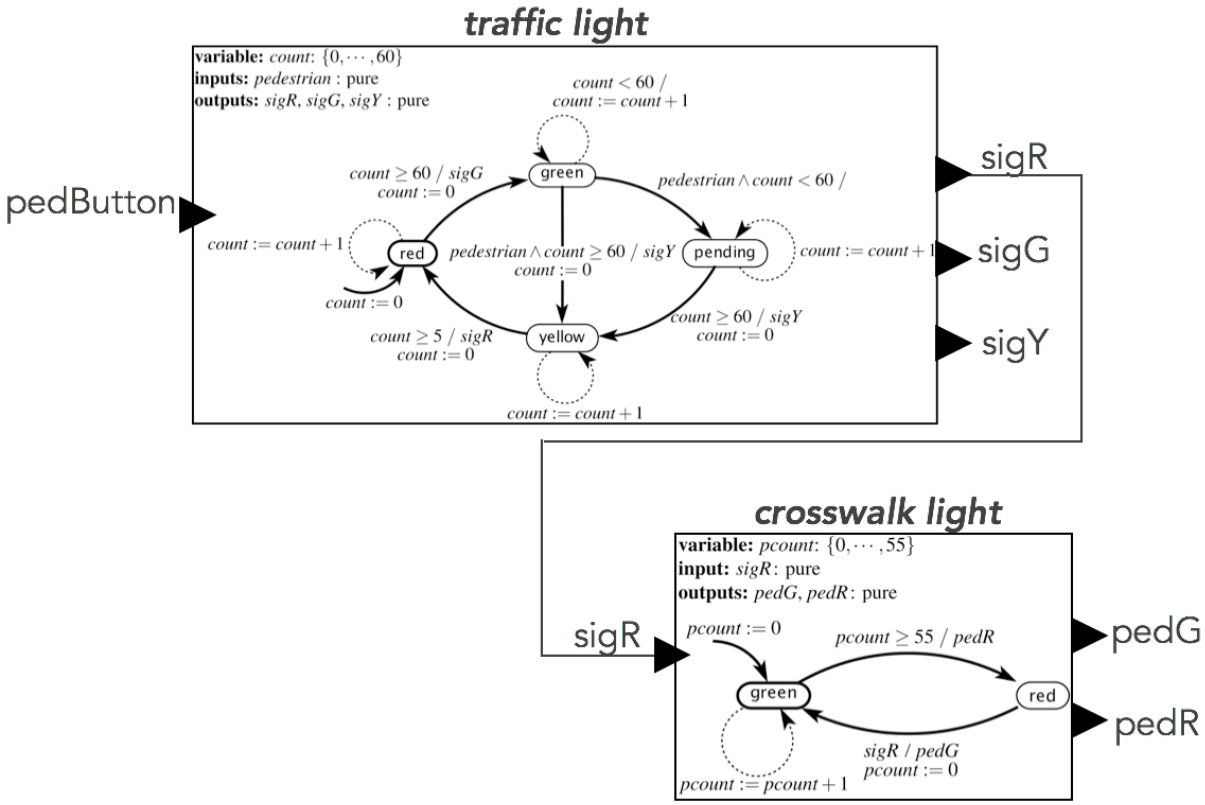
Four Vikings need to cross a damaged bridge in the middle of the night. The bridge can only carry two Vikings at a time and to find their way over the bridge (in either direction) the Vikings need to bring a torch. The Vikings have different speeds, and need 5, 10, 20 and 25 minutes respectively to cross the bridge in each direction. See course slides for additional discussion of the model.

Spend time to understand the model, and then use UPPAAL to answer the following questions:

1) Does an execution exist in which all four Vikings get across the bridge in less than or equal to 60 minutes? If yes, show the schedule of bridge crossings (e.g. as a Gantt chart or similar), and explain how you used UPPAAL to get the result.
2) Now assume that there is one more Viking who needs to cross this bridge and her time to cross the bridge is 20 minutes in each direction. What is the shortest time for all five Vikings to cross the bridge? Explain how you modified the system model to realize this change and explain what queries you used to find the shortest possible crossing time. Submit your modified model with queries as filename “a2.xml”
3) Building on question 2), if the bridge can carry three Vikings at a time, what is the shortest time for the five Vikings to cross the bridge? Explain how you modified the system model to realize this change and explain what queries you’ve used to find the shortest possible crossing time. Submit your modified model and queries as “a3.xml”

Part B (70 points)

Now that you have gotten some practice with UPPAAL, it is time to design and verify your own version of a traffic light controller and crosswalk light controller. We previously considered a discrete representation of the same system (shown below) in class, and now you should map it to timed automata using clock variables instead of discrete counts for tracking elapsed time. The traffic light model interacts with the crosswalk light model by asserting a signal sigR when the traffic light turns red.



1) In UPPAAL model the traffic light and crosswalk light system as timed automata. In this part of the question, the pedButton signal that causes the traffic light to (after some time) turn red is unconnected, so your model must include an additional component that generates this signal non-deterministically with no constraints (i.e. it models that pedButton can be pushed at any time). Test your system to gain confidence that it works. Explain what tests you performed. Include a table with the clock variables and communication channels used in your system.

2) For each of the following properties, which we describe as English sentences, explain whether your system should satisfy or violate the property. Then map the sentence to an appropriate reachability query, use UPPAAL to check it, and report the result. Provide each query in your report. Name the model and queries for this question as "b2.xml" and include this file with the submission so we can check it. For some properties you may find it helpful to add additional clocks that keep track of intervals between certain events but don't influence system behavior.
    a) "It is possible for the traffic and crosswalk lights to be green at the same time."
    b) "When the traffic light enters the green (non-pending) location, it will always change to another location within 90 seconds."
    c) "When the traffic light enters the pending location, it will always change to another location within 90 seconds."
    d) "When the crosswalk light is red, the traffic light is always green."
    e) "The traffic light never remains yellow for 20 seconds"

3) Now, add another component to the system that models a pedestrian. The pedestrian should have signal pedG as input coming from the crosswalk light, and should produce output pedButton going to the traffic light. The pedestrian model will replace the component you were previously using to generate the pedButton signal non-deterministically. The pedestrian model should have an idle location, a waiting location, and a crossing location. When transitioning from idle to waiting, the pedButton signal should be asserted to indicate that a request to cross has been made. The pedestrian should not start crossing until the pedG signal occurs. The pedestrian should spend a fixed amount of time in the crossing state, and that time must be a parameter of the model so you can easily change it, and we can easily test it; the name of that parameter must be pedCrossTime. Because the system makes certain assumptions about how quickly a pedestrian crosses the street, the system may be safe or unsafe depending on the value of the pedCrossTime parameter.
    a) Describe a bad condition that might occur in the model if the pedestrian spends too much time crossing the street.
    b) Which range of values for pedCrossTime allow the bad condition to occur, causing the system to be unsafe? Support your answer by using reachability queries to check that the bad condition can occur for some values of pedCrossTime, but cannot occur for other values of pedCrossTime.
    c) With pedCrossTime set to the largest safe value, name the model file and query as "b3.xml" and submit this file with the report. Since this is largest safe value, increasing pedCrossTime by 1 must change the result of the reachability query. We will test this.

4) [Note: this step is only for 2-person groups] Now change the model to be more robust, such that it is safe regardless of how long it takes the pedestrian to cross. Explain your change. Using your modified model, repeat your query from the prior step using a value of pedCrossTime that was unsafe in the previous step, and show that it is now safe after your change. You are allowed to change the traffic light and/or pedestrian light components in the new model if you find that helpful. Simulate and/or use other queries as well to gain confidence that your system works correctly. With pedCrossTime set to be a value that was unsafe in prior step, but is now safe, name the model file and query as "b4.xml" and submit this file with the report.