Aidan Chin SPIREID: 3380332 & Donivan Saville SPIREID: 33889341

A.1) Does an execution exist in which all four Vikings get across the bridge in less than or equal to 60 minutes? If yes, provide the schedule of bridge crossings, and explain how you used UPPAAL to get the result. Provide a schedule of the crossings as a table in your report. Each row of the table should correspond to one trip across the bridge. The columns of the table should indicate:

1. Which Viking or Vikings cross the bridge on the trip. You can identify them by their crossing times.
2. The start time of the trip.
3. Whether the trip starts from the safe or unsafe side of the bridge.

Yes, there exists a state where all 4 vikings cross safely under 60 minutes
To find this I enabled traces in the Diagnostic Traces: Shortest in the options tab and ran the Query listed above, this generated a trace in the simulation tab that I could manually step through as needed

Start of Trip	Viking(s) on trip	Start Location of Trip
Time >= 0	Viking 1	unsafe
Time >= 0	Viking 2	unsafe
Time >= 10	Viking 1	safe
Time >= 15	Viking 3	unsafe
Time >= 15	Viking 4	unsafe
Time >= 40	Viking 2	safe
Time >= 50	Viking 2	unsafe
Time >= 50	Viking 1	unsafe

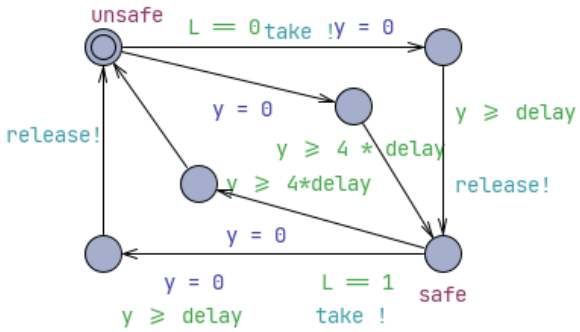
A.2) Now add a fifth Viking that needs 40 minutes to cross the bridge in each direction. What is the shortest time for all five Vikings to get across the bridge? Explain how you modified the system model to realize this change and explain what queries you used to find the shortest possible crossing time. Submit your modified model with queries as filename `<lastname>_a2.xml` (for example, my file would be called `holcomb_a2.xml`). As before, include a table to show the schedule of crossings.

I added a fifth that takes 40 minutes, again using the shortest trace I can find the minimum amount of time to take to cross, I find that it takes at least 100 minutes for everyone to cross, to verify this I also added 2 more verifications where time < 100 which is unsatisfiable, and another where time <= 100 which is satisfied

Start of Trip	Viking(s) on trip	Start Location of Trip
Time >= 0	Viking 1	unsafe
Time >= 0	Viking 2	unsafe
Time >= 10	Viking 1	safe
Time >= 15	Viking 1	unsafe
Time >= 15	Viking 3	unsafe
Time >= 35	Viking 1	safe
Time >= 40	Viking 4	unsafe
Time >= 40	Viking 5	unsafe
Time >= 80	Viking 2	safe
Time >= 90	Viking 1	unsafe
Time >= 90	Viking 2	unsafe

A.3) Building on question A.2, now consider a scenario in which each of the five Vikings has the option to reach the safe side by talking a longer path through the valley below, instead of crossing the bridge. Crossing through the valley does not require a torch. It takes each Viking 4 times longer to cross the valley than it does for them to cross the bridge. The bridge can still carry two Vikings, as before. There is no limit to the number of Vikings that can cross through the valley. In this modified scenario, what is the shortest time for all five Vikings to reach the safe side? Include in your report an image showing what modifications you've made to the model to answer this question, explain the modifications in your report, and explain what queries you've used to find the shortest possible crossing time. As before, include a table to show the schedule of crossings, but now your table must have a 4th column indicating whether each crossing is by bridge or valley. Submit your modified model and queries as `<lastname>_a3.xml`

This is the soldier model:



The difference I made was to include a way there and back through the valley, with the same type of guards and updates as the bridge but without the torch, I also multiplied the delay by 4 to get the longer time.
My query is the exact same as the one before:

The shortest time is now only 65 instead of 100 minutes

Start of Trip	Viking(s) on trip	Start Location of Trip	Path taking
Time >= 0	Viking 1	unsafe	valley
Time >= 0	Viking 2	unsafe	valley
Time >= 0	Viking 4	unsafe	bridge
Time >= 0	Viking 5	unsafe	bridge
Time >= 40	Viking 1	safe	bridge
Time >= 45	Viking 1	unsafe	bridge
Time >= 45	Viking 3	unsafe	bridge

B.1) For properties (b) through (i) in the table below, given as English sentences, explain whether the discrete system will satisfy the property or not. To do this, you should first devise a reachability check that could determine whether the property is true or false for the system; you are free to decide whether reaching the target state would show that the corresponding property is true for the system, or false. Then use your understanding of the model to predict the outcome and give a brief justification of how the target can be reached, or why it cannot be reached. Note that you are solving this problem with intuition and are not using tools to perform a reachability check on the discrete model. For some of the properties, you might have to include extra states or variables to track information needed for checking the property.

(a) It is impossible for the traffic light to be in the pending state 70 seconds after initial condition
(b) It is possible for the traffic and crosswalk lights to be green at the same time.
(c) The traffic light will never stay in the green (non-pending) location longer than 90 seconds
(d) The traffic light will never stay in the pending location for 55 seconds or longer.
(e) The traffic light will never stay in the pending location for 65 seconds or longer.
(f) It is possible that crosswalk light is red when the traffic light is green
(g) Whenever the crosswalk light is red, the traffic light must be green.
(h) The 3rd occurrence of sigG must not happen within the first 300 seconds after initial condition
(i) The 3rd occurrence of sigG must not happen within the first 400 seconds after initial condition

For demonstration purposes, here is an example of a suitable answer for condition (a):

Reachability check: I would add a variable called globalCount that initializes to 0 and increments in every reaction. I would check reachability of "globalCount == 70 and traffic light == pending". If that state is reachable, then property (a) is false.

Expected outcome: The property is false. The state can be reached by pushing pedButton right after the traffic light reaches "green".

b.)

Reachability check: I would check if trafficlight == green %% crosswalklight == green can be true at the same time

Expected outcome: False: The system is designed so that when the traffic light is green the crosswalk light has to be red immediately because of the sigR signal. So therefore they can both not be green at the same time.

c.)

Reachability check: I would track how long the traffic light stays in the green state and if longer than 90 seconds it fails

Expected outcome: False: The traffic light can remain green for exactly 120 seconds because it can go to the pending state for an additional 60 seconds and if the button isn't pressed it will remain green.

d.)

Reachability check: I would track how long the traffic light spends in the pending state and see if the time ever exceeds 55 seconds or more

Expected outcome: False: If the pedestrian presses the button late in the cycle it is possible for the pending state to last longer than 55 seconds.

e.)

Reachability check: I would track how long the traffic light spends in the pending state and see if the time ever exceeds 65 seconds or more

Expected outcome: True: Even if the pedestrian presses the button at the very last moment in the cycle it will still not exceed 65 seconds

f.)

Reachability check: I would check if trafficlight == green %% crosswalklight == red can be true at the same time

Expected outcome: True: I would expect this to be true because that would mean pedestrians can not cross when cars are crossing the road

g.)

Reachability check: I would check if trafficlight != green %% crosswalklight == red is reachable

Expected outcome: False: The traffic light could be yellow or in the pending state when the crosswalk light is red, especially if the button has not been pressed yet.

h.)

Reachability check: I would track the occurrences of sigG and check if the 3rd occurrence happens before 300 seconds

Expected outcome: False: Assuming that the traffic light cycles happens about every 120 seconds than the 3rd occurrence could happen before 300 seconds

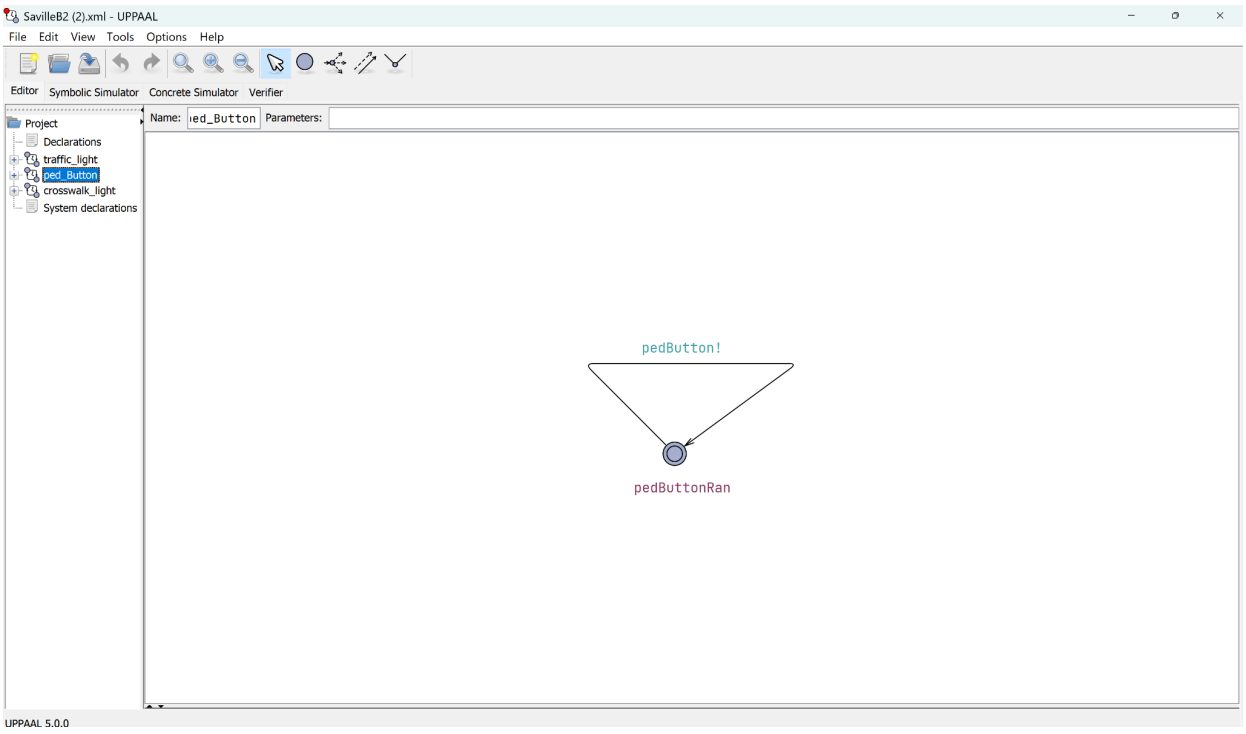
i.)

Reachability check: I would track the occurrences of sigG and check if the 3rd occurrence happens before 400 seconds

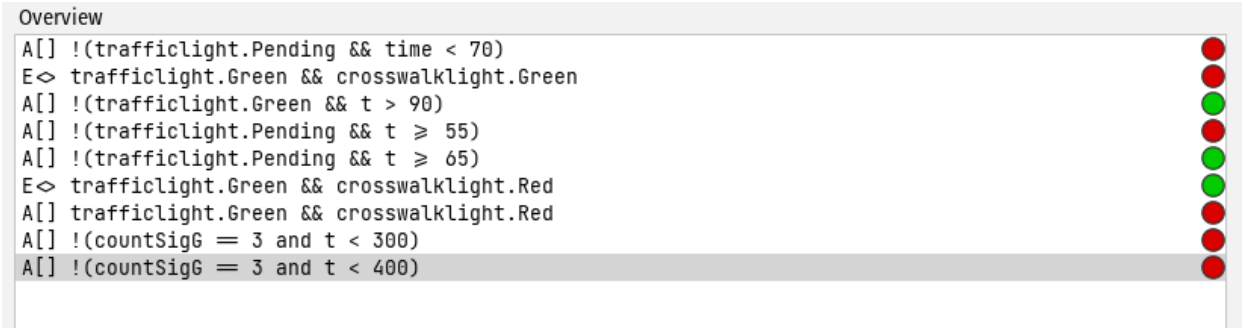
Expected outcome: True: Depending on pedestrian button presses delays and light cycles it could be possible that it occurs after 400 seconds

B.2) In UPPAAL model the traffic light and crosswalk light system as timed automata. Test your system (e.g. using concrete simulation or simple reachability checks) to gain confidence that it works as intended. You will then test it more formally in the next step.

- Your implementation must use timed automata and therefore will have clock variables to track time and must not have discrete count variables that increment once per second to track elapsed time.
- Remember that enabled jumps in timed automata are not compulsory unless there are also location invariants that force the jump to occur. This is different from the discrete model, which will always make a jump if at least one guard evaluates to true.
- The inputs and outputs of the traffic light and crosswalk light must be connected to communication channels. Every communication channel requires synchronized transmit and receive events (e.g. `take!` and `take?` in the Viking model):
  - Channel *`sigR`* will have traffic light and crosswalk light as transmitter and receiver.
  - Since the *`pedButton`* input to the traffic light is unconnected, your model must include an additional component to generate it non-deterministically (to model that *`pedButton`* could be pushed at any time) and transmit it to the traffic light over the channel. Include in your report a screenshot of the component you created that non-deterministically generates *`pedButton`*.
  - The unused outputs (*`sigG`*, *`sigY`*, *`pedG`*, *`pedR`*) will have traffic light or crosswalk light as transmitter, so you will need to add corresponding receivers for these communication channels and ensure they are always ready to receive, so as not to block jumps within the transmitting module.
- Some edges in the discrete model use an input in the guard condition and assert an output when taking the guarded transition. UPPAAL will not allow a single jump to interact with multiple communication channels. You can avoid this problem by adding an urgent location to split a single jump into two jumps with no time elapsing between them.



B.3) Map each condition from the table in B.1 to an appropriate query that can be checked in UPPAAL on your model from B.2, and report the result. Provide in your report a screenshot of the verifier window that shows all queries and whether they are reachable (UPPAAL indicates this by adding red or green circles next to each query once it is checked). Name the model, with all queries included, as `<lastname>_b3.xml` and include this file with the submission so we can check it. Some properties may require modifications in your model; for example, to keep track of intervals between events, or to count the number of times that an event occurs. You are allowed to make these modifications but must use a single model version for all queries since you will submit the model and queries as a single file. Include a query for property (a) even though we've already described above how it can be checked.



B.4) Now add another component to the system that models a pedestrian trying to cross the crosswalk. The pedestrian's input should be *pedG* from the crosswalk light, and the pedestrian's output should be *pedButton* going to the traffic light. The pedestrian model will therefore replace the component you were previously using to generate and transmit *pedButton* and also replace the component you were using to receive *pedG*. The pedestrian model should have an **idle** initial location, a **waiting** location, and a **crossing** location. When transitioning from idle to waiting, the *pedButton* output should get asserted to indicate that a request to cross the street has been made. The pedestrian starts crossing when the *pedG* input occurs. The pedestrian should spend a fixed amount of time in the crossing location, 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 already 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.

- Describe the bad condition that can occur if the pedestrian crosses the street too slowly.
- What 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 show that the bad condition can occur for some values of `pedCrossTime` but cannot occur for other values of `pedCrossTime`
- With `pedCrossTime` set to the smallest unsafe value, name the file as `<lastname>_b4.xml` and submit it with the report. Since this is the smallest unsafe value, decreasing `pedCrossTime` by 1 must change the result of the reachability query. We will test this.

- If the pedestrian were to cross the street too slowly they are likely to get hit by an oncoming vehicle
- Any value greater than 54 allows the bad condition to occur that is because they can cross the street at the same time the light is green and I used this reachability query E<> (pedestrian.Walking && trafficlight.Green) and set my pedCrossTime to 55 and it was green meaning they can be true at the same time while at 54 it is red meaning they will never happen at the same time.

Submit your assignment by uploading to Canvas your typed report, and the files from steps **A.2**, **A.3**, **B.3**, **B.4**, and (for ECE622 students) **B.5**. Each model will include your lastname as part of the filename. Be sure to close and then re-open each file to confirm that the file you are submitting is the file that you intend to submit, and that it includes the appropriate queries. If you work as partners, both student names should be included on the report, but only one student will submit the files for the group.