Specification and Verification in Uppaal

José Proença
jose.proenca@fc.up.pt

System verification – 2024/2025

To do

Develop the requested Uppaal specification and requirements, and produce a report in PDF, including screenshots of the requested timed automata.

What to submit

The *PDF report* **and** the developed *Uppaal models*.

How to submit using git

1. Use the git repository used for the first assignment on mCRL2 and extend it.
2. Make sure that `jose.proenca@fc.up.pt` was added as a member of the group (read-permissions are enough).
3. Include all the files to be submitted in the repository.

Note that **all students should push commits.**

Deadline

4 Jan (Saturday)

Motor controller in a Railway system

A railway company produces signalling systems that are used in critical systems. These must provide enough evidence over their reliability and correctness over time to comply with the heavy certification processes.

This assignment is a simplification of recent use-case of an European project, depicted in fig. 1. In this use-case a critical motor that rotates left and right interacts with a controller. This controller runs on a resource-constrained device with a real-time OS. In turn, a remote dashboard sends commands and receives updates to/from the controller. The goal of this assignment is to analyse the behaviour mainly of the controller, interacting with the motor and the dashboard.

Controller behaviour The overall behaviour of the controller is summarised in fig. 2. Initially the controller is idle, where it must remain for at least 1000ms to perform some bootstrap self-tests. It can

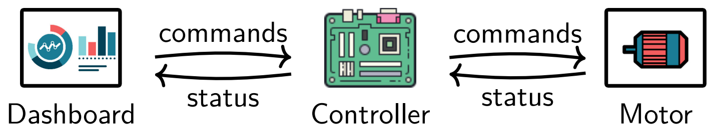

Figure 1: Architecture of the motor controller system under verification

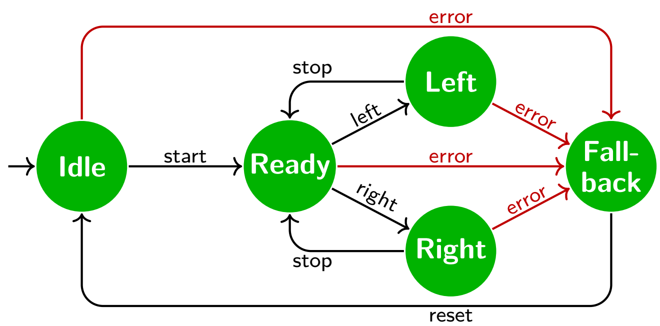

Figure 2: General behaviour of the controller component

then receive a start command to become ready to actuate. Once it is ready, it can receive a left (resp. right) command from the dashboard, which will trigger an instruction move-left (resp. move-right) to the motor.

The controller then checks every 400ms if the motor reached the movement limit or if it detected some error. This check is made using a shared register between the motor and the controller. If the controller detects that the limit is reached between 4000ms and 5000ms after the move instruction is sent, it becomes idle and notifies the dashboard. Otherwise it raises an error (more info on this below) and goes to a fallback state. If the motor is misbehaving, it should not take more than 6000ms to raise an error since the move-instruction is sent. While in a fallback state, the controller waits for the dashboard to send a reset command to become idle again.

Raising errors Everytime an error is raised by the controller an error message must be sent to the dashboard and a stop message must be sent to the motor.

Three components with global constants The controller's core behaviour is described above. The dashboard and the motor will be also modelled as two dedicated timed automata. Your model should be parameterised by (at least) the following global constants:

- Minimum and maximum time at the idle and start phase;
- Minimum and maximum time to expect the limit to be reached;
- Periodicity to read the status of the the motor;

Exercise 1. Model this system as an UPPAAL model and **submit** it to the repository. We would like to have multiple versions: one for each **scenario**. Each scenario will include a controller and/or a motor with different behaviour. Model at least two scenarios:

- when both dashboard and motor behave well, i.e., never lead the system to a fallback state;
- when the motor misbehaves, i.e., leads the system to a fallback state.

You can model more scenarios to capture different cases when the system behaves well or misbehaves. **Describe** this UPPAAL model and the 3 scenarios in your report, **justifying** clearly your decisions (assumptions and abstractions) made in this modelling exercise.

Exercise 2. For safety reasons, these systems need redundancy to reduce the chances of failure. A safer system is a variation of the model above that uses 2 controllers for redundancy, both interacting with the dashboard and the motor.

While in states `Ready`, `Left`, and `Right` (c.f. fig. 2, the two controllers check if they are consistent. More specifically, every 100ms each of the two controllers should check if the other controller is in the same state, using shared variables. After 3 failed attempts this controller should raise an error and go to its fallback state.

Furthermore, the motor uses two different channels (or shared variables) to send information to the two controllers (which may be inconsistent in a faulty scenario).

Create an updated model of this system and **submit** it to the repository. As before, model at least two concrete scenarios, of a successful and an unsuccessful execution. **Describe** this UPPAAL model and the 3 scenarios in your report, **justifying** clearly your decisions (assumptions and abstractions) made in this modelling exercise.

Exercise 3. Use UPPAAL's CTL logic to express and verify properties.

3.1. Formulate 4 properties of one of the systems above based on their descriptions. If you want, you can make new assumptions not described above, making it explicit what is new.

3.2. Express a property in UPPAAL's CTL logic for each item below. Fix a particular set of parameters (c.f. the problem description), and say if each property holds or not for your model (or if it takes too much time), and explain why.

1. It is possible to move the engine 3 time to the left in less than 15000ms;
2. Whenever a fallback state is reached, it must take at least 6000ms until the motor can turn again to the left.
3. Until a scenario is finished, the system does not deadlock.