Assignment 2: Modelling communication within the Kobuki Robot

José Proença

Arquitectura e Cálculo – 2016/2017

To do: Develop Uppaal models and Reo connectors as requested, and write a report using LaTeX. This report should include screenshots of the requested time automata, diagrams (hand-made or with the Reo tools) with the requested Reo connectors, and properties that you verify.

To submit: The *report* in PDF, **and** the developed *Uppaal models* (for Ex. 2). Send by email a unique zip file "`ac2-UPP-N.zip`" (Ex. 1) and another "`ac2-Reo-N.zip`" (Ex. 2) to `jose@proenca.org`, where `N` is your group number.

Demo: Explained in Exercise 3.

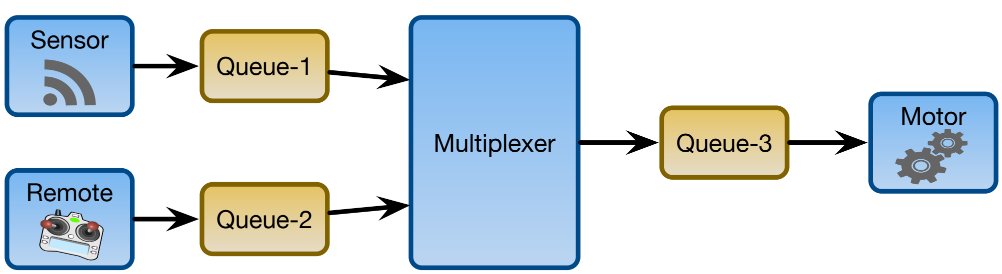
Deadline Exercise 1: 27 Apr 2017 @ 14h (Thursday)

Deadline Exercise 2: 31 May 2017 @ 23h59 (Wednesday)

Real-time modelling

Exercise 1. In this exercise you will model a part of the software running in the Kuboki robot (`http://kobuki.yujinrobot.com/`), focusing on time aspects, and using a simplified version of a recent publication (`http://jose.proenca.org/papers/ros-formalise17.pdf`).

The overall architecture is depicted below. A sensor component detects obstacles, and sends a stop message whenever that happens. Concurrently, a remote component sends movement messages to control the robot. These two messages are forward by a multiplexer component to the motors, giving higher priority to the sensor messages.



1.1. Model this architecture using 7 timed automata, one for each component and queue, taking into account the following restrictions.

- Both the sensor and the remote have a parameter `Rate` indicating the exact frequency at which they produce messages.
- Each queue i has a parameter $\texttt{Size}_i$ that describes the maximum size of the queue.

- Trying to enqueue a value on a full queue is allowed, resulting in an *overflow* (the data is lost).
- The multiplexer has a `Lock` integer parameter, and handles priority as follows.
  - The sensor messages have higher priority remote messages.
  - After any sensor message is received, the multiplexer will discard any remote message received in the next `Lock` time units.
  - Received messages that are not discarded are forward immediately to the motor component.

1.2. Suggest a small modification to your model **with a timelock**, and another modification **with zeno behaviour**. Explain why these are undesirable models.

1.3. Express three properties using UPPAAL's CTL, one for each of the following items.

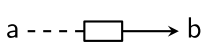
- ϕ_1 – the queue of the first publisher cannot *overflow*;
- ϕ_2 – the queue of the subscriber cannot *overflow*;
- ϕ_3 – every message sent by the 2^{nd} publisher must be received within N time units.

1.4. Find, for each property, a combination of parameters `Rate`$_{\text{Sensor}}$, `Rate`$_{\text{Remote}}$, `Size`$_i$, and `Lock` that **satisfies them**. Similarly find, for each property, a combination of parameters that **rejects them**.

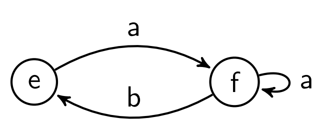
Coordination with Reo

Exercise 2. You will now model the same architectural scenario as above using Reo connectors, and abstracting away from time. For that, assume that the sensor, the remote, and the motor are components that are always ready to send or to receive data, and the rest will be encoded as a Reo connector.

2.1. Model the connector between the sensor, the remote, and the motor. For simplicity, assume the queues have exactly size one (can store at most one value), and use the Lossy-fifo channel defined below (depicted on the left and defined as an automata on the right). **Draw the resulting connector.**



Lossy-fifo



2.2. Build a variation of this connector, replacing the Lossy-fifo ( ) by the composed connector , and using a sensor and remote component definitions that produce only 2 data values each. Using this variation and with the help of the Eclipse plug-in, build the mCRL2 specification of this connector and visualise the resulting LTS. Make sure you include in the LTS the actions regarding data being send and received from/to the components. **Show the mCRL2 specification and the LTS**.

2.3. Write 2 desired properties in mCRL2 that hold in the connector modelled in mCRL2.

2.4. Try to understand why replacing the Lossy-fifo by the new composed connector can produce undesired behaviour. **Write 1 property** (or more) that shows a desired property that captures this undesired behaviour, i.e., which will not hold in the variation built in Exercise 2.2 but would make sense to hold in the connector you proposed before in Exercise 2.1.

Demo

Exercise 3. Present your UPPAAL and Reo models, discuss your design choices, and show desired properties that hold and properties that do not hold. If possible, show variations of your models and explain advantages and disadvantages.