Lecture

Topics:

- Discussion of Week 1 Exercises
- Introduction to *timed automata*: syntax and semantics
- Timed automata in UPPAAL
- Modelling with timed automata

Readings

1. Read section 9.1 (pages 673-698) in [BKch9TA].
2. Read sections 1–3 (skipping 2.2 and 2.3 for now) in [UPPAAL].

Both of these texts can be found on DTU Learn.

Exercise 1: Skew and Jitter

In this exercise you are going to model two undesired, but common, phenomena of real-time systems. In both cases we are considering a process which (ideally) is supposed to repeatedly emit a signal with a constant period of 10 time units.

1. If the physical clock of a system is not running at the correct rate, the system may suffer from *skew*, where the signals may get more and more displaced from their correct nominal times.

   Make a UPPAAL model of the behaviour of the process if the clock rate may deviate from the nominal rate with ± 10 % (changing dynamically in within these bounds).

   What is the maximum and mininum distance which may be observed between the first and the last of five consecutive signals?

2. Even if the clock is running precisely, the output of a process may be displaced due to internal data handling. The delay of the output with respect to the nominal times is called *jitter*. Jitter is commonly experienced in the end-user presentation of video and/or audio streams and may reduce the quality significantly.

   Make a UPPAAL model of the process where the output may suffer jitter up to 1 time unit.

   What is the maximum and mininum distance which may be observed between the first and the last of five consecutive signals?

Hint: Use a channel for the signal event and introduce an observer who accepts signals at any time.

Exercise 2: Pelican Crossing

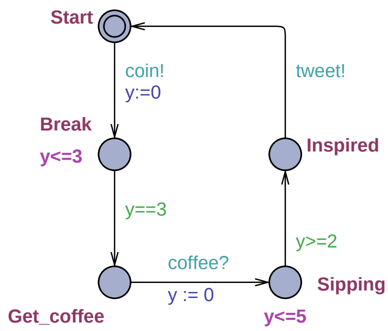
A road with heavy traffic has a pedestrian crossing with a traffic light which may stop the traffic on request of the pedestrians (aka. a *pelican crossing*). The light should stop the traffic for a period of 15 seconds to enable all pedestrians to cross. The traffic stop is requested by pressing a button.

To avoid a total traffic stop, an overall requirement is that traffic must not be stopped for more than 20 % of the time.

Make a UPPAAL model of the pelican light control (and some pedestrians to operate it) such that pedestrians are eventually allowed to cross and the traffic not stopped too much. (Amber light periods may be ignored.)

Exercise 3: Coffee Machine

In this exercise we will design the control of a Machine which will serve a coffee-craving Person, and a Follower who will check that the Person produces *tweets* (on Twitter) often enough. The Person repeatedly inserts a coin, takes a short break, gets back to extract the coffee, sips it for a while and finally emits a tweet. Each of these activities takes a certain amount of time as specified by the following model of the Person:



[Can be found in the model file `Coffee0.xml`]

The machine takes varying time to brew the coffee after a coin has been inserted, and will time-out if the coffee has not been taken before a certain upper time-limit. As a requirement we want the overall behaviour to ensure a minimum flow of tweets within the system. Solve the following tasks in UPPAAL:

1. Model the Machine such that it will react on the Person inserting a coin, then brew coffee for $[t_{min}, t_{max}]$ minutes, and finally offer the coffee to the Person, or discard the coffee if it has not been taken after t_{lim} minutes (once brewed). Assume that the Person will try to insert coins and retract the coffee as soon as possible.

   Hint: Pass the constants t_{min} etc. as parameters to be determined by the system instantiation, so that they can be easily changed.

2. Create a Follower that checks how often the Person produces a new tweek and who might get disappointed if at any time more than 10 minutes elapse between two consecutive tweets. Assume that the Person will emit his tweets as soon as they are ready.

3. Analyse the behaviour of the system, and try to determine (by calculations) the constraints that the parameters t_{min}, t_{max}, and t_{lim} must satisfy for the system to work without deadlocks or disappointments.

4. (Optional) Verify your results by formulating suitable invariants and show that they hold/do not hold as expected, when varying the parameters.