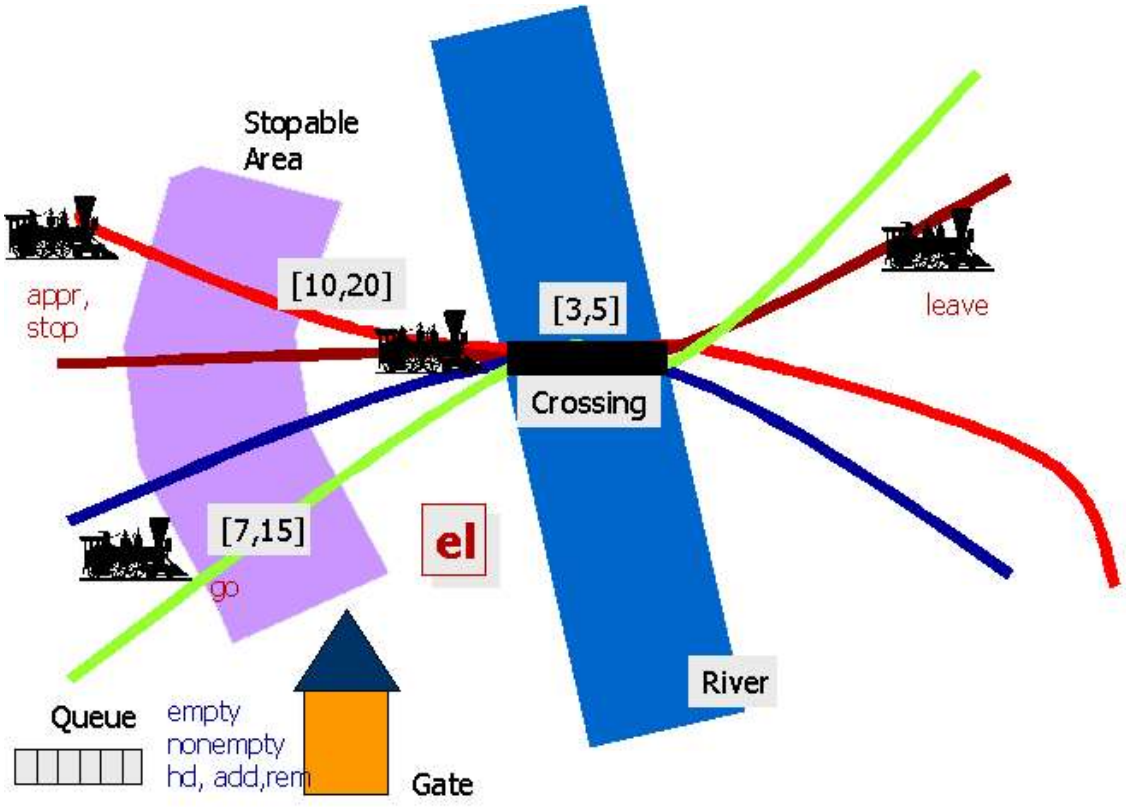
TuToR 2017 Exercises for UPPAAL

Exercises

Exercise 1 (Train-Gate Error Correction and Modification)

You may find the correct version of the train-gate example that follows with the 4.1.19 version here.

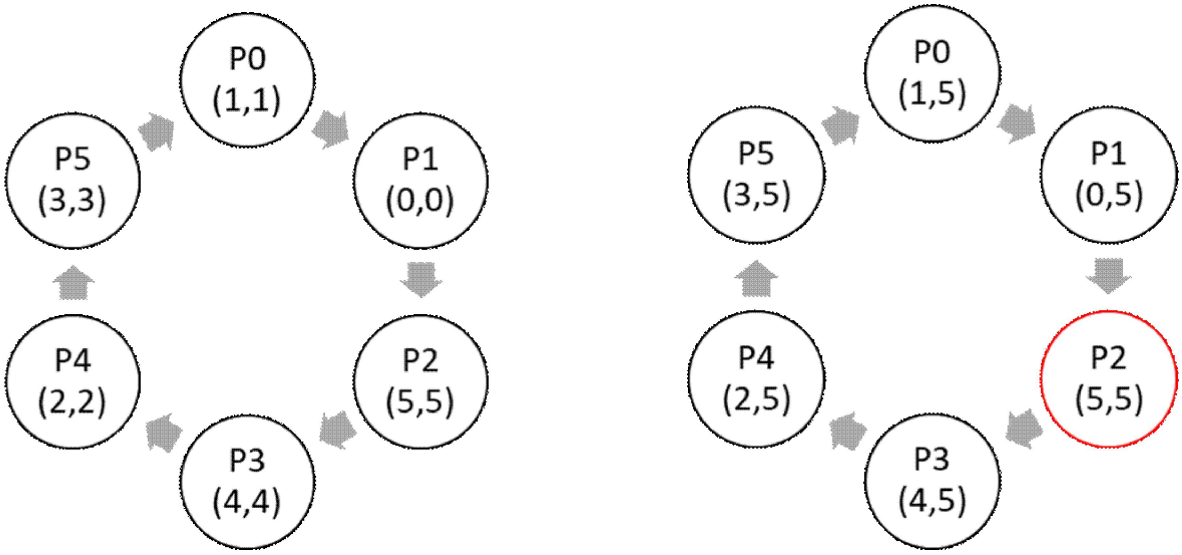
The following contains an erroneous version of the train-gate example. Here is the appropriate query-file.



- Use UPPAAL (simulation, verification) to pin-point and explain the error(s).
- Having corrected the errors what is the minimum time one can guarantee between a train requesting access to the crossing and actually getting there?

- Now we assume that trains will have a priority according to their *id*. You can find a version of the train-gate example with the new scheduling-policy of the Gate here. What can be said about the correctness of the system. Investigate performance properties of the system (e.g. try identify the amount of time the various trains may be consecutively stopped within the first 1000 minutes). In which train would you prefer to be a passenger?
- Change the model so that a train can only make a new approach after some time *MIN* after having left the crossing. What is the minimum value of *MIN* that will ensure that no train is starved even under fixed priorities? For each train what is the minimum time one can guarantee between a train *id* is attempting to approach the crossing and actually leaving it?

Exercise 2 (Leader Election)



We consider leader election in a ring topology. We assume that each node of the ring has a *unique identifier* and a unique *priority.* The object of the algorithm is to have all nodes identifying the (node with the) maximum priority (above this is node 2 which has priority 5). We consider the LCR algorithm (Lann, Chang and Roberts), where each process sends its priority around the ring. When a process receives an incoming priority, it compares that priority to its own, it keeps passing the priority; if it is less than its own, it discards the incoming priority; if it is equal to its own, the process declares itself the leader.

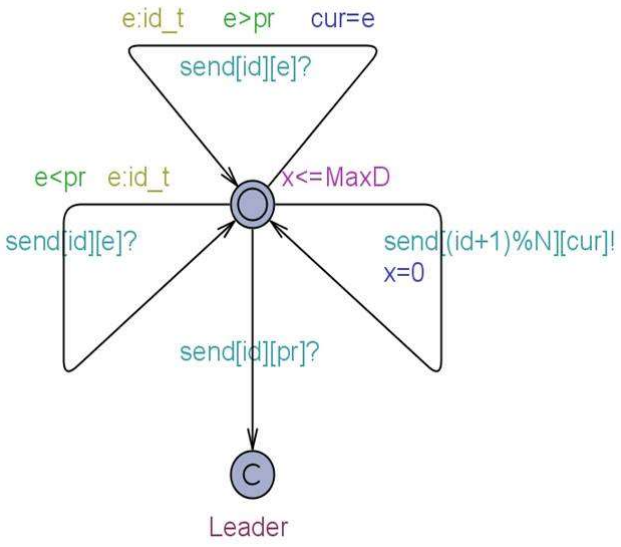
We model the behavior of a process as a timed automaton (template) in UPPAAL as illustrated left. The model and an associated query may be found here (simpleleader3.xml, simpleleader3.q).

Q1. Simulate your model to ensure that it corresponds to the informally given explanation above.

Q2. Verify that a leader eventually be elected (what does that mean)? What is the maximum amount of time before the leader is elected?

Q3. What is the maximum number of priorities sent in the network before the leader is elected?

Q4. What is the expected time before a leader is elected? What is the probability that the leader is elected before 10 time-units.



Q5. What is the expected number of priorities sent before the leader is elected?

Q6. Modify the model so that priorities are only sent (before MaxD time-units) with a certain probability (say 70%). Investigate the effect of this on the properties (e.g. those above) of the protocol.

Q7. Investigate what happens if a node completely fails (e.g. by the transmission probability being 0%).

Exercise 3 (Job shop scheduling)

Jobshop scheduling problems involves a number of jobs -- J0,..,Jn -- to be treated by a number of machines -- M0,..Mk. In this particular variant each job Ji needs to be treated exactly once by each machine and in a particular order and for a particular time-duration. The problem is to find an time-assignment indicating at which time Ji should be treated by Mj in a way such that the prefered sequence of Ji is respected and so that a machine Mj is never used by more than one job at a time. The challenge is to identify the minimum Makespan, i.e. time-assignment which the minimal total duration.

	Sport	Economy	Local News	Comic Stip
Kim	2. 5 min	4. 1 min	3. 3 min	1. 10 min
Jüri	1. 10 min	2. 20 min	3. 1 min	4. 1 min
Jan	4. 1 min	1. 13 min	3. 11 min	2. 11 min
Wang	1. 1 min	2. 1 min	3. 1 min	4. 1 min

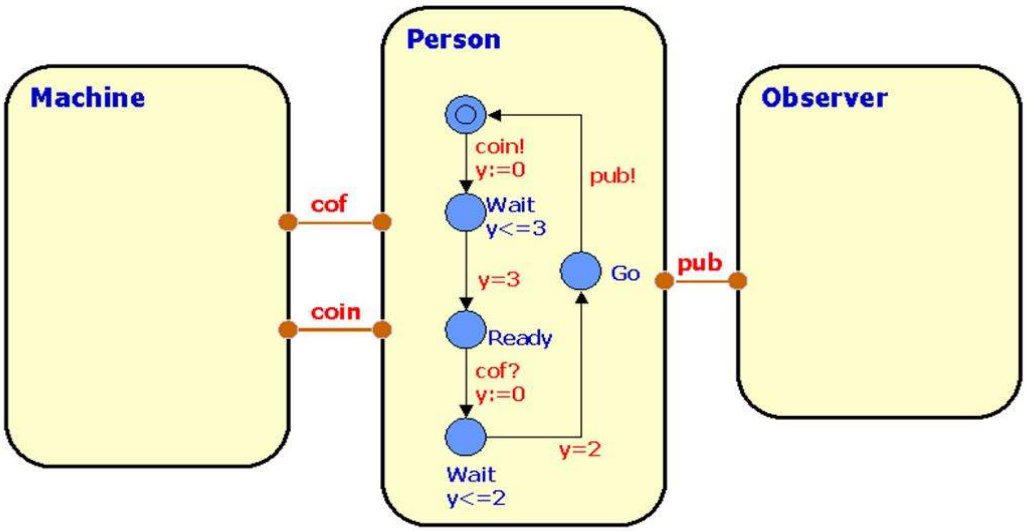
Consider the particular Jobshop scheduling problem above, where 4 persons (Kim, Juri, Wang and Jan = the Jobs) needs to read 4 sections (the Machines) of a single news paper. Make a UPPAAL model of this instance of the jobshop. You may want use the following UPPAAL skeleton model containing a generic Job-template (see here) or full model (see here). Use the model checking engine of UPPAAL to determine the minimum make-span, i.e. the minimum time until all persons has completed reading all sections. Identify also the minimum time for Kim to finish reading his sections, and the minimum time for Wang to finish his sections. What is the minimum time to have both Kim and Jan finishing reading their sections. Try to illustrate using informative Gantt charts.

	Sport	Economy	Local News	Comic Stip
Kim	2. 3-6 min	4. 1-5 min	3. 3-6 min	1. 10-11 min
Jüri	1. 8-10 min	2. 20-23 min	3. 1-6 min	4. 1-12 min
Jan	4. 1-3 min	1. 13-20 min	3. 11-23 min	2. 11-13 min
Wang	1. 1-3 min	2. 1-5 min	3. 1-3 min	4. 1-4 min

Consider the Jobshop scheduling problem with the time required for a given person to read a certain being given by a uniform distribution over a given interval (see table above). Modify the Job-shop scheduling model that you made in the previous part of this exercise (see the skeleton here) to take these timing intervals into account (in the template). Add exponential rates for each person (proposed rates are: Kim: 5, Juri: 2, Jan: 6, Wang: 1) defining the distributions by which the person will start to read a section. Now use the SMC engine to determine the following interesting quantities. What is the probability that all persons will have completed his reading before time 200 – have a look at the cumulative probability distributions. What is the probability that Kim finishes before Wang within 200 minutes? What is the probability that Wang finishes before Kim within 200 minutes? Use hypothesis testing to check for each of these unknown probabilities whether it is larger than 0.5? Also compare these two probabilities?

Finally equip the model with local clocks for each person in order to estimate the individual expected completion times. Also add stop-watches for each person in order to estimate the expected total waiting time (that is time when a person is NOT reading)

Exercise 4 (Coffee Machine)



In this exercise you are asked to design the control of a **Machine** (the control program) which will serve a coffee craving **Person** (the environment). As you can see above the person repeatedly (tries to) insert a coin, (tries to) extract coffee after which (s)he will make a publication. Between each action the person requires a suitable time-delay before being ready to participate in the next one.

The machine takes some time for brewing the coffee and will time-out if coffee has not been taken before a certain upper time-limit.

As a requirement we want the overall behaviour to ensure that the indicated **Observer** experiences a constant flow of publications from the system. In particular we want the Observer to complain if at any time more than 8 time-units elapses between two consecutive publications. Model the Machine and Observer in UPPAAL and analyse the behaviour of the system. Try to determine the maximum brewing time allowed by the Machine in order that the above requirement is met.

An initial sketch of a UPPAAL can be found here.

Exercise 5 (Gossiping Girls)

Model and analyze the following gossiping girls problem in UPPAAL. A number of girls initially know one distinct secret each. Each girl has access to a phone which can be used to call another girl to share their secrets. Each time two girls talk to each other they always exchange all secrets with each other (thus after the phone call they both know all secrets they knew together before the phone call). The girls can communicate only in pairs (no conference calls) but it is possible that different pairs of girls talk concurrently. For all of the tasks below you should consider the following three scenarios:

- Scenario 1: all girls may directly call any other girl, thus telephone network is a total graph.
- Scenario 2: only a single call in the entire network can take place at a time
- Scenario 3: the girls are organized in a *linear chain (with a first and last girl)* restricting calls to be made between neighboring persons.

Your tasks are as follows:

- Model the above three scenarios as a network of UPPAAL timed automata (using channels, clocks, timed automata templates, C-code, …)
- Use UPPAAL to find the minimal number of phone calls needed for four girls to know all secrets in each of the three scenarios.
- Refine your model so that each phone call lasts exactly 60 seconds (for simplicity this time duration is irrelevant of the number of exchanged secrets). Find the minimum time needed to solve the gossiping girls problem for four girls in each of the three scenarios.
- Experiment with the UPPAAL search options breath-first and depth-first search, random-depth-first search, upper and lower approximations, and with the diagnostic trace settings fastest and shortest. Try to solve the problem for five girls.
- Assume that the time of a call between two girls is uniformly distributed between 40 and 60 seconds. Also assume that a girl after having completed a call makes a new call after a 20 and 30 seconds accoriding to a uniform distriubution. Also assume that the call is made to a (uniformly) randomly choosen (other) girl. Using UPPAAL SMC to determine the expected time until all girls know all secrets. Also what is the probability that all girls know all secrets before 150, 200, 250, 300, 400 seconds.

Hint:

- Design a single template for all girls.
- For each girl, you may choose to remember the currently known secrets either in a local array of booleans or using an integer variable (use a binary encoding such that if a girl knows the secrets of e.g. girls 1 and 3 but does not know the secrets of girls 2 and 4, the value in the integer variable will be (0101) binary = 5; you might find useful the operation | for a bitwise OR).