Modeling and analysis using Uppaal

Below an exercise in modeling and analysis using the model checker Uppaal.

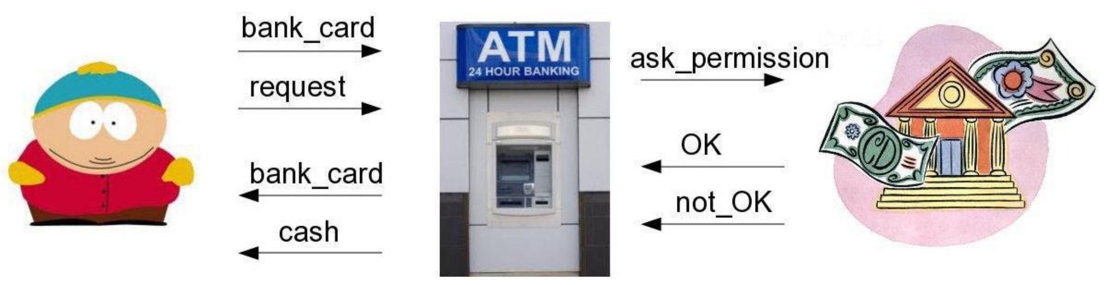
Uppaal is free to download and very easy to install and use. The exercise can be done by high school or undergraduate students without any prior knowledge, after a short explanation of what a state diagram is, and a demo that walks through the basic features of Uppaal. (The explanation should cover Uppaal's notation `?` and `!` for input and output actions on channels, the notion of deadlock, explain how you edit the state diagrams, and show the simulator and the verifier, including how the verifier will produce counter-examples that you can step through in the simulator.)

Model checking

Model checking is a technique that can be used in the design and analysis of dynamic systems. A model checker is a computer program that rapidly and cleverly searches through all the possible states of a system to look for problems. Effectively, you are googling for potential problems.

A model checker does not work with the real system, but a *model* of the system - hence the name. Uppaal uses *state diagrams* as models. Apart from a diagram describing the system we want to analyse, we also need to describe some desired properties. These properties are called *queries*.

Exercise: an ATM system



Install Uppaal and then download & save the files below:

- atm.xml
- atm.q

Open these files in Uppaal: atm.xlm using File:Open System and atm.q using File:Import Queries. The file atm.xml provides a model of an ATM (Automated Teller Machine, or cashpoint), a customer, and a bank. To keep things simple, the bank only has one ATM and one customer. There customer, called Eric carries Eric.cash_in_pocket euro in his Wallet. The ATM has with ATM.in_till euro in till. The Bank is the back-end system of the bank, and keeps track of Eric's balance in Bank.balance.
Eric interacts with the ATM to withdraw cash from the machine. The ATM in turn communicates with the Bank, to make sure that the bank correctly keeps track of the balance of Eric's bank account.
Intially, Eric has no cash in his pocket. You can check this in Uppaal by clicking on the system left of Eric, and then on clicking on Declarations. Eric has a balance of 80 euro in his bank account, and the ATM has 200 euro in the till.

The file atm.q expresses two properties of the systems: namely that Eric always owns 80 euro (since the model does not include the possibility for him to spend any money) and that the system should not deadlock.

Question 1 : Use the verifier to check the properties. Try to improve the model until both properties are true. For the first property you have to change the bank. For the second property you have to change Eric. (What should Eric do if the ATM does not pay out cash but does return his bank card?)
Uppaal tip: under the menu Options:Diagnostic Trace select the option Shortest to let the verifier generates a counterexample in the simulator.

Question 2 : Change the ATM so that initially it only has 30 euro in the till instead of 200.
There is still something in the model of the ATM that is not realistic: with this lower amount of money in the till something can happen in the model which would be impossible in reality. Try to figure out what this is. You can do that by simulating the model for a while, or by letting Uppaal do a random simulation. (Hint: Eric wants to do some shopping and needs a lot of cash.)

Change the model of the ATM to improve this.

Once you have done this, add a query to the verifier to check that the problem has been avoided. The query should express the obvious reality check for the ATM. Of course, the adapted model still should not deadlock!