Internet Engineering

Paweł Głuchowski

Information Systems Analysis

Temporal Logic in System Analysis

(corrected version, 2013)

Project co-financed by European Union within European Social Fund

Instruction to the 1st laboratory assignment of Temporal Logic in System Analysis

Subject: LTL logic and system properties description.

During this assignment you will practise:

- drawing a representation of given LTL formulas in linear time structures,
- writing LTL formulas representing given linear time structures,
- specifying properties of a given program in LTL.

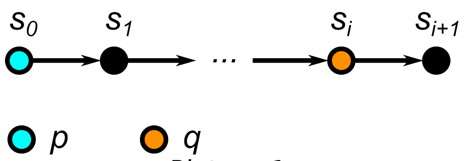
Tutorial for exercises 1 and 2:

Picture 1 (below) presents a linear time structure.

The structure consists of big points and arrows joining them together in a line. The points are described by symbols like s_i and by colours. There is also a triple of dots between some arrows.

Every point represents a state (moment) of the time structure. Above every point there is the state's symbol and its index (e.g. the state no. 7 would be s_7). If a point is black, then its state is labelled with the value *true* (meaning that everything is possible then but *false*). Another colour represents a formula which is true in this point's state. This labelling function is explained under the time structure. Usually, instead of using colours, we write the formulas by the states.

Every arrow represents a transition between states. If an arrow connects two states, then these states represent the order of two moments, where there is no moment in-between. If there is a triple of dots between two arrows, then we could place some number of other states instead of the triple, which all would be the same as the state, from which the left arrow goes out.



Picture 1.

Picture 1 presents a formula: $p \wedge Fq$, which is true in the states s_0. Let $i \geq 1$ (in states' indexing). If i=1, then clearly the states s_1 and s_i are the same state and are labelled with $true \wedge q \equiv q$. If i=2, then there is only one state s_1, labelled with *true*, between states s_0 and s_i. Otherwise, there is some grater number of states between s_1 and s_i, all labelled like s_1, that is, with *true.* It is important always to declare relation between the states connected by the arrows with dots between them.

Exercises

Exercise 1.

Draw linear time structures representing given LTL formulas, where each formula is true in the state s_0. Place symbols of formulas p and q by these states, in which these formulas are true. The states without formulas' symbols will be understood as labelled by *„true"* (do not write *true* there).

a) Draw a linear time structure representing the following LTL formula:

$X(p \wedge G(p \Rightarrow Xq \wedge XXXp))$

b) Draw a linear time structure representing the following LTL formula:

$\neg p \wedge pUq \wedge FGq$

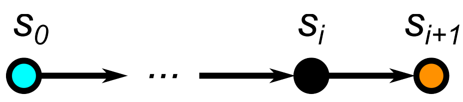
c) Draw a linear time structure representing the following LTL formula:

$pU(Xq) \wedge p$

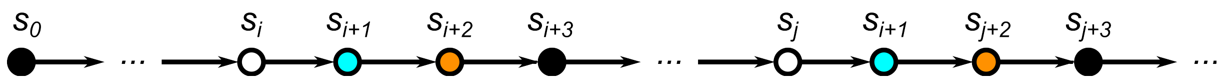
Exercise 2.

Write LTL formulas with symbols: p, q and any logic connectives, to represent given linear time structures for the following labelling function: b (blue), o (orange), w (white), *true* (black). Do not write *„true"* in the formulas.

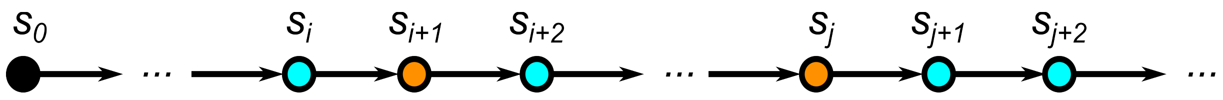
a) Write an LTL formula representing the following linear time structure for i>0:



b) Write an LTL formula representing the following linear time structure for i>0 and j≥i+3:



c) Write an LTL formula representing the following linear time structure for i≥0 and j>i+2:



Exercise 3.

Specify as many properties as you can for a chosen communication protocol.

Choose one of the following communication protocols that you know the best: FTP, SSH, Telnet and TLS and do the following:

- Try to describe as many aspects of using the protocol, as you can, in spoken language and in LTL formulas, where a property or event etc. will be symbolised by a letter. Make sure first that you know these protocols.
- Try to find the most important properties that must be satisfied by the protocol's users to ensure a correct use of the protocol. Protocol users are programs rather than people.
- If it is possible, for every formula draw a linear time structure.

What you are about to do now, will be useful in some of the following assignment, when you will use a similar temporal language (CTL) to model the use of protocols and verify its properties automatically.

Subject: Modelling a system as automata and specifying its properties in LTL.

During these two assignments you will practise:

- using a tool for modelling, specifying and verifying – UPPAAL,
- modelling given single-process systems as deterministic finite state automata,
- specifying properties of the models in LTL.

Tutorial for the exercises:

In the beginning you need the "UPPAAL 4.0 Small Tutorial", which you may download from the official UPPAAL site at www.uppaal.com.

(direct address: www.it.uu.se/research/group/darts/uppaal/small_tutorial.pdf).

To become familiar with the UPPAAL application, please follow all instructions from chapters 3.1 and 3.2 from the UPPAAL tutorial. It will take some time, but they will teach you how to use the tool. It is no problem, if you do not understand everything the automata from the tutorial do. After next assignments you certainly will.

Whenever you find some difficulty in modelling, simulating or verifying your systems in UPPAAL, feel free to check other chapters of this manual (or other manuals published there).

Exercises

Exercise 1.

Model given single-process systems as deterministic finite state automata.

To built an automaton for a single-process system, find out what states can these systems be in, and what transitions between them are possible. Remember: transitions must be chosen deterministically, so describe the transitions with proper guards and variable updates.

After you model them, simulate their run to check, whether they work as intended.

a) Model a digital combination lock, that unlocks after choosing a sequence: 1,7,8,3. The lock reads all 10 digits from 0 to 9 infinitely, until the sequence is read. The correct sequence may be chosen at any time, that is, after any number of digits not constituting the sequence. When the sequence is received, the lock unlocks and remains unlocked forever.

Hint: Use committed states, which are left at the very same moment, that they are entered.

b) Model a ticket machine, which receives coins, calculates their value and returns a ticket. There is only one kind of a ticket, hence a user initiates the machine by inserting a coin. He may insert coins as long as the ticket is not returned. The machine does not return the rest, if overpaid. Whenever the summary value of coins is greater or equal to the ticket's price, the machine returns the ticket.

Notice, that there are many different coins. A user may use at least one coin, and as many coins as he wants.

Exercise 2.

Specify properties of the models from exercise 1 in LTL logic.

Try to find as many properties of your models, as you can. Especially, you should consider the following kinds of properties:

- a deadlock of the system is never possible, unless it reaches a final state (if there is any);
- if some states must be reached to correctly run the process, they will eventually be reached;
- if some states of the system must not be reached to correctly run the process, they will never be reached;
- all correct actions of the system may be performed, that is all correct states may be reached; etc.

Write the properties as LTL formulas without nesting the temporal operators (UPPAAL has some restrictions yet), and verify all the properties with UPPAAL verifier. If the verification is not successful, correct your model (automaton) and/or your temporal formulas.

Hint: Instead of temporal operator G write 'A[]', instead of F write 'E<>', instead of $\Rightarrow$ write 'imply', instead of $\wedge$ write 'and', and instead of $\vee$ write 'or'.

Hint: To check the possibility of a deadlock write 'A[] not deadlock'.

Subject: Modelling a system as automata and specifying its properties in CTL.

During these two assignments you will practise:

- modelling a given multiprogram concurrent system as indeterministic finite state automata,
- specifying properties of the model in CTL,
- modelling and verifying systems that use a communication protocol (from the first laboratory of temporal logic in system analysis).

Tutorial for the exercises (with yet another exercise inside!):

The goal for each of the exercises is: to build a correct model of a system using more than one automaton, to specify its properties (e.g. safety, reachability) in CTL logic and to verify them.

Whenever you find a difficulty in modelling, simulating or verifying your systems in UPPAAL, fill free to check the UPPAAL manuals from www.uppaal.com.

The following is a simple example (you will have to do much more during the laboratories) of a system of some programs, that communicate with each other. The model is made in UPPAAL modeller. Then some properties of the model are specified in CTL logic, which allows to verify them by the UPPAAL verifier.

Example.

Let us model a simple chat system consisting of two kinds of programs: a user application U and a server S. There are three instances of user applications: U(1), U(2) and U(3) and one instance of the server: S. We will model the logging in and out only, where the following property must be satisfied (apart from the fact that the system must work): a user may log in or out to the server.

We will create two templates: U for the user application, and S for the server. Write in the project's system declarations the following: *system U, S;*

We start by creating a template for a user application. Name it 'U' and write next to the name a parameter: *const N id*

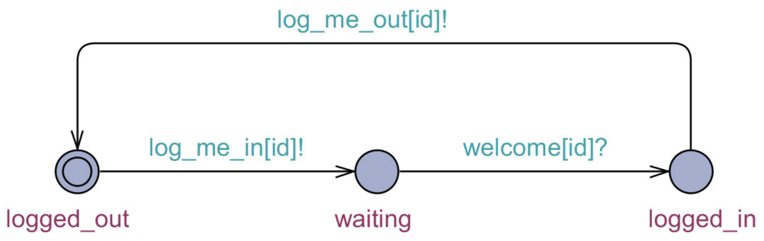
Now we want to have three user applications numbered from 0 to 2, so we have to write the following in the project's declarations: *typedef int[0,2] N;*

Next build an automaton depicted below in Picture 1, where: names of states are purple, the state *logged_out* is initial, communicates sent (!) and received (?) through channels are light-blue, and *id* is a user's number. To use the channels, write the following in the project's declarations:

chan log_me_in[N], log_me_out[N];

broadcast chan welcome[N], bye[N];

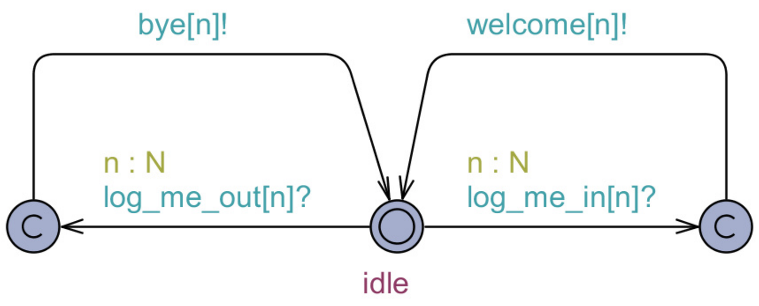
A *chan* is a normal one-to-one communication channel. A *broadcast chan* is a one-to-all communication channel.



Picture 1. *User interface 'U' template*

Now we may build the server automaton. Create a new template, name it 'S', and built an automaton depicted below in Picture 2. The server might have one state only (*idle*), but to perform more than one operation of the same kind (e.g. handling of two communicate), we need one transition for each. This is possible by using the committed states, marked 'c'.

In sent and received communicates, the selection of a 'U' instance is done by the transition's select property (brown colour). For example, if the server receives a communicate *log_me_in* from U(1), then it stores the user's number in 'n' (here, n:=1), which allows the server to send a communicate *welcome[n]!* welcoming the logging-in of U(n), that is U(1).



Picture 2. *Server 'S'*

Our system does not fully handle the bye communicate broadcasting yet. How would you model that? Do it as <u>yet another exercise</u>.

For our system we can specify at least the following properties and verify them:

- The system cannot be deadlocked: *A[] not deadlock*
- Every user may log in, e.g.: *E<> forall (i : N) U(i).logged_in*

etc.

Notice, that nesting of temporal operators is not allowed in UPPAAL yet, which narrows our opportunities of the verification process. Other tools may not have this restriction, but you would have to learn their automata declaration languages, instead of building them graphically.

THE DEVELOPMENT OF THE POTENTIAL AND ACADEMIC PROGRAMMES OF WROCŁAW UNIVERSITY OF TECHNOLOGY

Exercises

Exercise 1.

Model a system that uses the communication protocol you described in the exercise 3 of the first assignment of temporal logic in system analysis.

You have already specified the system and its properties in the spoken language. Now model the system as automata, and write the properties as CTL formulas. Verify as many properties, as possible, in UPPAAL. Try to manually "verify" the rest in simulation.

Summary – you shall:

- build an indeterministic automata model with clocks, channels etc. and simulate it to ensure the proper use of the system's protocol;
- formulate as many safety and reachability properties of the system in CTL logic, as you can; formulating other properties would also be nice;
- automatically verify as many properties, as possible; verification must be successful, according to the protocol.
- try to verify all the other properties by simulation.

If the verification is not successful, correct your automata and/or temporal formulas.

The teacher may give you other orders, so that you learned everything, that the laboratories allow.

Hint: To write the formula $AG(p \Rightarrow AFq)$ write p --> q

Project co-financed by European Union within European Social Fund