System validation, 2IW26

Jan Friso Groote (J.F.Groote@tue.nl, HG 6.75, 040-247 5003).
http://www.win.tue.nl/~jfg/educ/2IW26/herfst2011/overzicht.html

The purpose of this course is to learn how to specify behaviour of systems and to experience the design of a system where you can prove that the behaviour is correct. So, you will learn how to formally specify requirements and to prove (or disprove) them on the behaviour. With a practical assignment you will experience how to apply the techniques in practice.

The first block of the semester will be dedicated to learn the basics of the specification language mCRL2 and to use it as a manual specification and verification tool. In the first block there will be 2x2hours of lectures per week, on wednesdays and fridays. The second block is devoted to an assignment, which must be finished before the examination period at the end of the semester. The goal of the assignment is to apply the techniques and tools to the design of a small distributed and/or embedded system. The purpose is to design this system such that it is proven to comply with all the requirements which must have been formulated in advance.

In the second block there will be a few lectures on fridays. Directly after the first block there will be an examination. There is one possibility to retry this exam, at the end of the semester.

The marks for the exam and the assignment both contribute equally to the final score. The final mark is the average rounded to a whole number in the ordinary way (0.5 is rounded up). Failing for the course means that both the exam and the assignment must be redone next year. It is possible to commence with the assignment without having passed the exam.

Literature

The course material consists of

- J.F. Groote and M.R. Mousavi. Modelling and analysis of communicating systems. Lecture notes. 2011. Chapters 1, 2, 3, 4, 5, 6, 7, 9 (not 9.7 and 9.8.3), 10, 11, 12.
- See www.mcrl2.org for the tools, manual pages etc.

The lecture notes by Groote and Mousavi can be obtained at the office of the secretary (Tineke van de Bosch, 247 5010, HG 6.74). Otherwise, the documents can be downloaded from the website (`www.win.tue.nl/~jfg/educ/2IW26/herfst2011/overzicht.html`). The exam will cover the indicated parts of the lecture notes as well as everything said during the lectures.

Assignment

The assignment consists of designing a controller for a small distributed and/or embedded system. Below a suggestion for such a system can be found. It deals with a controller for an ASML waferstepper. But it is possible to design any embedded controller or distributed algorithm provided you obtain approval by the supervisor of your assignment. The assignment can be carried out in groups of one to four persons.

Carrying out the assignment consists of executing the following steps:

1. Identify in words global requirements for the whole system. Typical requirements are 'packets are never stored at places that are occupied'. These requirements are initially to be described in natural language.
2. Identify the interactions that are relevant for your system. Describe clearly but compactly the meaning of each interaction in words.
3. Translate the global requirements in terms of these interactions.

4. Describe a compact architecture of the structure of the system. It is required that the controller has at least three different parallel components.

5. Describe the behaviour of all controllers in the architecture using mCRL2.

6. Verify using the toolset that all requirements given in item 3 above are valid for the design in mCRL2.

The assignment must be documented in a technical report that covers all items above. This report must be a concise technical account of the system and must be written such that from it the requirements, action interface, architecture and behavioural design can be easily understood. It must also be clear how the requirements are verified, in such a way that this can easily be redone without consulting any of the authors of the report.

ASML is designing a new generation wafer steppers. These allow for even smaller components to be put on wafers by using ultra violet instead of visible light for the process of copying masks onto the silicon. A problem of ultra violet light is that it is absorbed by the atmosphere. Therefore, this new waferstepper can only function in an extreme vacuum.

This new generation wafer steppers is equipped with a number of vacuum sluices through which wafers enter and leave the machine. It is of utmost importance that the vacuum in the machine is always maintained. The first reason is that the machine is equipped with costly high quality lenses that cannot stand normal atmospheric conditions for a longer time. These may require replacement when exposed to air too long. The second reason is that it requires days to restore the required extreme vacuum in the machine, causing a loss in production time.

Current plans are that there will be two sluices both with two doors that give entrance to a low vacuum chamber. From this there is a single door that will give access to the high vacuum chamber. All rooms are equiped with a vacuum sensor. All doors may be equiped with a sensor to feel that they are fully open and properly closed. It is possible that more sluices will be added to increase throughput.

It is intended to design a software/hardware interface that can be instructed to accept a unprocessed wafer, that indicates through which sluice it may enter, monitors the state inside the machine and allows processed wafers to leave the machine. During operation the machine must monitor all sensors and steer all actuators to guarantee proper operation of the machine. The specification of this hardware software interface must specify the operation of the machine in such a way that elementary safety properties are met. A very obvious correctness criterion is that it never happens that both doors of the same sluice are unintentionally open. It is the first task of the software designers to provide a list of 'requirements' that the system must satisfy. These requirements must be presented in such a way that it is obvious to those using this component that it will function as required.

The system must be fault tolerant. Especially in the case of hardware problems the system may never violate any of the requirements. In particular it is possible that doors do not open or close due to motor malfunctioning, that it is not properly detected what the position of doors is, that vacuum sensors yield false readings, etc. It is important to define which readings sensors do deliver, when they malfunction. Furthermore, the system behaviour must be analysed to see that under no series of allowed malfunctionings any of the requirements are violated. An important difficulty is that it is very important that the machine can make as much progress as possible. If one sluice cannot be used anymore, all movements in and out the machine must be rerouted through the other sluice that does function well.

The system should also allow for maintenance. For instance one of the sluices can be taken out of service when it requires care of some sort. It must be possible to open the outer door of such a sluice while the other sluice is still in full operation. Another feature that the controller should have is that it can indicate when the machine is idle, namely when there are no wafers in the system anymore.

Tool set

See *www.mcrl2.org* and the webpage of the course.

Global time schedule

Below a global time schedule is indicated. Furthermore, there are exercises indicated, which can be treated during the lectures. Students are supposed to make these exercises before the lectures, and compare their answers with those of the lecturer. The exercises can be found in the lecture notes.

7/9-9-2011. Chapter 1. Chapter 2. Chapter 4. Transition systems, basic processes, process equivalences, conditional operator, time. Elementary reasoning with axioms. Exercises 2.2.2, 2.2.3, 2.3.2, 2.3.7, 2.3.8, 2.3.9, 2.5.5, 2.5.6 4.3.1, 4.3.2, 4.5.1, 4.5.2.

14/16-9-2011. Chapter 3. Chapter 4. Section 9.4. Appendix A. Abstract data types. Constructors. Built in data types, bool, quantifiers, pos, nat, int, real. list, set, bag, functions, structured type. Difference between $\approx$ and $=$. Predefined data types, induction. Exercise 3.1.1, 3.1.2, 3.1.3, 3.2.1, 3.2.3, 3.2.4, 3.3.1, 3.4.1, 3.5.1, 3.5.3, 3.5.4, 3.5.5.

21/23-9-2011. Section 4.6. Chapter 5. Section 9.6. Recursion. RSP. Proving recursive specifications equal. Parallel processes and hiding. Expansion law. Communication, multi-actions. Exercises. 4.6.1, 5.1.1, 5.2.1, 5.4.1, 9.6.4, 9.6.5, 9.6.6, 9.6.9.

28/30-9-2011. Chapter 6. The modal μ-calculus with data. Exercise 6.1.2, 6.2.1, 6.3.1, 6.3.2, 6.4.1, 6.4.2, 6.4.3.

5/7-10-2011. Chapter 9. Lambda calculus. Sum axioms. Sum elimination theorem. Precise proof system. Exercise 9.4.1, 9.4.2, 9.4.3, 9.5.2, 9.5.3, 9.5.4, 9.5.5.

12/14-10-2011. Chapter 10. Chapter 11. Linearisation of processes. CL-RSP, CL-RSP with invariants. Confluence and τ-priorisation. Exercise 10.1.4, 10.2.10, 10.2.11. Exercise 11.1.3, 11.1.4, 11.2.6.

19/21-10-2011. Chapter 12. Cones and foci theorem Exercise 12.1.3, 12.1.4, 12.1.5.

27-10-2011. Reserve.

31-10-2011. Exam (9:00-12:00). Closing date for registration: 23-10-2011.

16-11-2011. Assignment.

23-11-2011. The toolset and its philosophy.

30-11-2011. Reserve.

7/14/21-12-2011. Reserve.

11/18-1-2012. Reserve.

9-1-2012. Last date to hand in the pre-final report for the assignment.

20-1-2012. Last date to hand in the final report for the assignment.

26-1-2012. Exam (retry, 9:00-12:00). Closing date for registration 15-1-2012.