System validation, 2IW26

Jan Friso Groote (J.F.Groote@tue.nl, HG 6.75, 040-247 5003).
Mohammad Mousavi (M.R.Mousavi@tue.nl, HG 6.79, 040-247 2993).
`http://www.win.tue.nl/~jfg/educ/2IW26/herfst2010/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 thursdays. 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 wednesdays. 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.A. Reniers. Modelling and analysis of communicating systems. Lecture notes. 2010. Chapters 1, 2, 4, 5, 6, 7.1-7.1.4, 8 (not 8.8.3, 8.7), 9, 10, 11.
- See www.mcrl2.org for the tools, manual pages etc.

The lecture notes by Groote and Reniers 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/herfst2010/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 small packet storage system. 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.

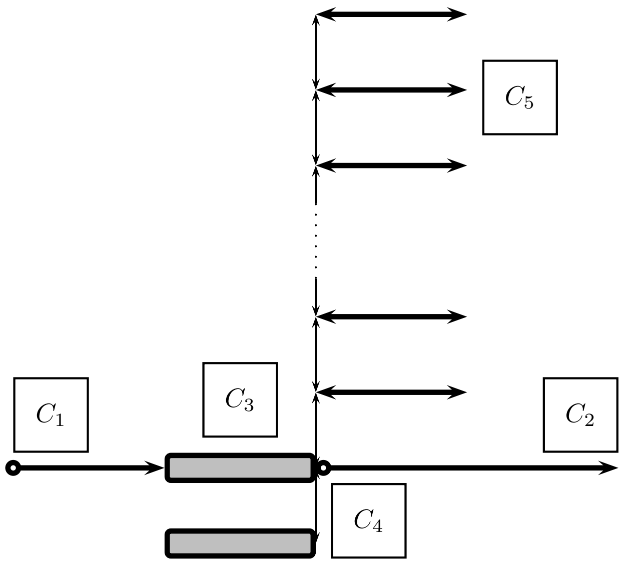

Figure 1: A temporary storage with two elevators

3. Translate the global requirements in terms of these interactions.
4. Describe a compact architecture of the structure of the system.
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.

The default assignment of this year consists of a small packet storage system (see figure 1). The assignment is inspired by an operational product manufactured by a company called VanDerLande, located in Veghel, appr. 20 km from Eindhoven (`www.vanderlande.nl`). This company produces luggage handling-, packet storage- and parcel distribution systems.

Packets arrive at the left (at C_1). Using a unidirectional conveyor belt packets are moved to the elevators (C_3 and C_4). With the two elevators they can be moved up and down and be stored in the rack (C_5). The lowest elevator can move to the level below the conveyor belt, but it cannot reach the highest rack. The upper elevator can reach the highest rack, but cannot reach the lowest position. The elevators are mounted above each other, which means that they can not reach the same position. Self-evidently, they cannot pass each other either. Each platform of the elevators can store at most one packet.

Each level at the rack can contain exactly one packet. When a platform reaches the right level, instructions must be given to the rack and the elevator to move a packet from or onto the elevator platform. This also holds for the conveyors. When a platform is at the same level as the conveyors, the platform and the conveyor hardware must be instructed to move the packet.

If a packet arrives at this mini warehouse (C_1), the system can decide to accept the packet, but it can also decide to leave it at the entrance if there is no space to store it. However, it is appreciated that an as large as possible number of packets can be stored, where if possible also the elevator platforms can be used as storage positions.

At the right (C_2) packets can be ordered which will then be delivered in the same order as they have been requested. It can be assumed that packets can always be accepted by whatever device requested the packets.

There are five controllers C_1, C_2, C_3, C_4 and C_5 that govern the behaviour of the system. The assignment is to model the behaviour of these controllers and prove that they cooperate well. Each controller is only allowed to have knowledge of its own device. So, only C_5 is allowed to know which levels of the rack are filled with which packets. And similarly, only C_3 knows to which level the upper elevator is travelling. The purpose of this is to make each device with its controller modular and therefore reusable in other contexts.

This means that if a packet is ordered at the controller C_2, it must request all the other controllers to find out whether a packet is present and where it is located. Subsequently, a plan must be negotiated to move the packet to the exit. Note that at the same time packets can enter the system, which must be stored simultaneously.

If you work on this assignment, you will find out that the description, although quite precise at first sight, still leaves quite a number of aspects undetermined. This is quite common, and by itself already a reason to make a formal description of the behaviour of the controllers. In cases where the behaviour is not well described in this text, you must make your own choice regarding the desired behaviour. When you feel that deviating from this text would yield a system with a nicer behaviour you are allowed to do so, provided you can defend your choice.

What is important is that if packets are present in the system, the system will always deliver them. Under no circumstance the system may deadlock or be able to only deliver the packets in predetermined orders. Furthermore, packets may not bump into each other. This also holds for the elevators.

In a realistic system, one may also have to deal with failing motors or other mishap, meaning that an elevator gets stuck or a storage level cannot be used. Sometimes, it is even the case that packets are removed from the storage rack or drop to the floor. In such cases the system must inform the outside world that something has gone wrong. It is not necessary to deal with these situations in this assignment. It is however important to realise that the current system is quite a simplification of those systems that occur in reality.

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.

1/2-9-2010. Chapter 1. Sections 2.1–2.3. Sections 4.1 and 4.3. Transition systems, basic processes, process equivalences, conditional operator, time. Elementary reasoning with axioms. Exercises 2.2.2, 2.3.2, 2.3.3, 2.3.4, 2.3.6, 2.3.9, 4.1.3, 4.1.4, 4.1.5, 4.3.3.

8/9-9-2010. Section 4.2. Appendix A. Section 8.4. Abstract data types. Constructors. Built in data types, bool, quantifiers, pos, nat, int, real. list, set, bag, functions, structured type. Difference between $\approx$ and $=$. Appendix A, induction. Exercise 4.2.1, 4.2.2, 4.2.3, 4.2.5, 4.2.6, 4.2.7, 4.2.8, 4.2.12, 4.2.14, 8.4.1.

15/16-9-2010. Section 4.1.5 (cont.), 7.1–7.1.4, 8.1–8.3, 8.5. Lambda calculus. Sum axioms. Sum elimination theorem. Precise proof system. Exercise 4.1.6, 8.4.2, 8.4.3, 8.5.2, 8.5.3, 8.5.4, 8.5.5.

22/23-9-2010. Section 4.1.6, 8.6. Recursion. RSP. Proving recursive specifications equal. Exercises. 4.1.7, 4.1.8, 8.6.4, 8.6.5, 8.6.6, 8.6.9.

29/30-9-2010. Section 2.4, 2.5, 4.4, 4.5, 8.8. Parallel processes and hiding. Expansion law. Communication, multi-actions. Exercise 2.5.4, 2.5.5, 2.5.6, 4.4.5, 4.5.1, 4.5.2, 8.8.1, 8.8.2.

6/7-9-2010. Chapter 9. Linearisation of processes. CL-RSP, CL-RSP with invariants. Exercise 9.1.4, 9.2.10, 9.2.12. Chapter 10. Confluence and τ-priorisation. Exercise 10.1.3, 10.1.4, 10.2.6.

13/14-10-2010. Chapter 5. The modal μ-calculus with data. Exercise 5.1.2, 5.2.1, 5.3.1, 5.3.2, 5.4.1, 5.5.1, 5.5.2.

20/21-10-2010. Chapter 11. Cones and foci theorem Exercise 11.1.3, 11.1.4, 11.1.5.

4-11-2010. Exam (14:00-17:00). Closing date for registration: 17-10-2010.

10-11-2010. Assignment. Process visualisation.

17-11-2010. The toolset and its philosophy.

24-11-2010. State space generation and term rewriting.

1/8/15-12-2010. Reserve.

3-1-2011. Last date to hand in the pre-final report for the assignment.

5/12-1-2011. Reserve.

14-1-2011. Last date to hand in the final report for the assignment.

17-1-2011. Exam (retry, 14:00-17:00). Closing date for registration 9-1-2011.