Prof. Dr. A. Podelski, Dr. B. Westphal
M. Vujinovic

Wintersemester 2015/16

Software Design, Modeling, and Analysis in UML

http://swt.informatik.uni-freiburg.de/teaching/WS2015-16/sdmauml

Exercise Sheet 6

Early submission: Monday, 2016-01-25, 12:00 Regular submission: Tuesday, 2016-01-26, 10:00

Exercise 1 (4/20 Points)

Consider the UML model given in Figure 1.

(i) Which behaviour is possible from (σ, ε) given by Figure 1(d)?
Assume that E and F are signals, but not environment signals. (2)

(ii) Enter the model into Rhapsody and simulate it.
How does the behaviour you observe in Rhapsody relate to your results from Task (i)? (2)

Hint: this is the essence of the bonus task from the last exercise sheet. So those who worked on that bonus task in a sense enjoy a double bonus...

Exercise 2 (3/20 Points)

Regarding the consistency of a UML model with OCL constraints, Slide 12 of Lecture 14 says "evaluate the constraint for each 'reasonable point'", i.e. for certain reasonable system configurations considered during model evolution.

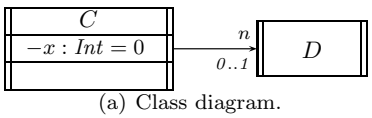
(i) Discuss 'reasonable points' using the state machine from Figure 2 assuming an object in state s_1 and stable with $x = 1$. Further assume that E and F are environment signals, i.e. they may occur at any point in time. Use the following OCL constraints:

a) context C inv : $x > 0$

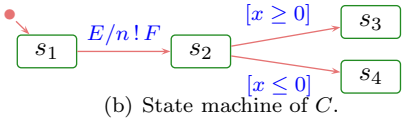
b) context C inv : $x \geq 0$

c) context C inv : $x \leq 1$

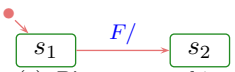
For each constraint, there is a choice of 'reasonable points' such that the constraint holds. What about the other constraints then? (2)



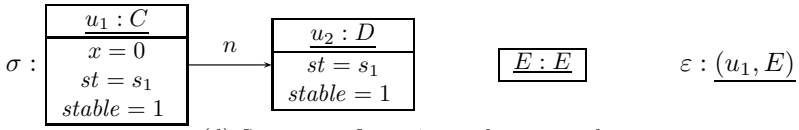
(a) Class diagram.



(b) State machine of C.



(c) D's state machine.



(d) System configuration and event pool.

Figure 1: Formal semantics vs. Rhapsody.

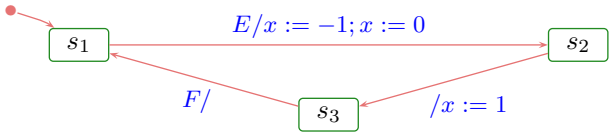


Figure 2: State machine for Exercise 2.

(ii) In the lecture, we choose to consider the system configurations before (and after) one step as 'reasonable point'.

Why is this a good choice, i.e. are there transformations of constraints (or the model) which hold even when using the choice of 'reasonable points' taken in the lecture. (1)

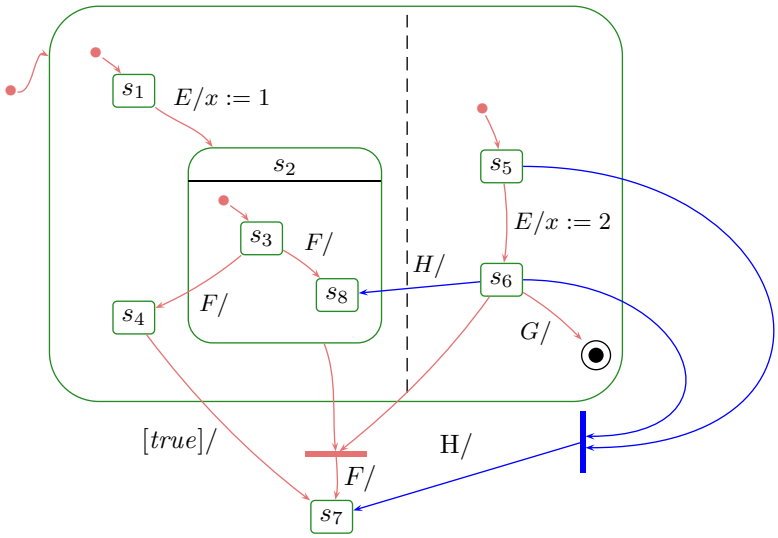


Figure 3: Hierarchical State Machine.

Exercise 3 (6/20 Points)

Consider the hierarchical state machine shown in Figure 3.

(i) Provide the mathematical representation of the diagram in Figure 3. (1)

(ii) Point out an example of a basic, composite, OR-, AND-, and final state, and one pseudo-state. (1)

(iii) Are the blue transitions legal or not? Briefly explain why. (1)

Do not consider non-legal transitions in the following tasks.

(iv) Assume the considered state machine belongs to class C and we have one instance u of C in σ and a global, shared ether

$$\varepsilon = \underline{(u, E_1)}.(u, E_2)$$

where $E_1 \in \mathscr{D}(E)$ and $E_2 \in \mathscr{D}(F)$.

Which behaviour is possible from (σ, ε)?

Explain briefly, which transition(s) are taken why. (1)

(v) Which behaviour is possible from (σ, ε) if $E_2 \in \mathscr{D}(G)$? Will the object be alive at the end of all possible evolutions? (1)

(vi) Which behaviour is possible from (σ, ε) if $E_2 \in \mathscr{D}(H)$? (1)

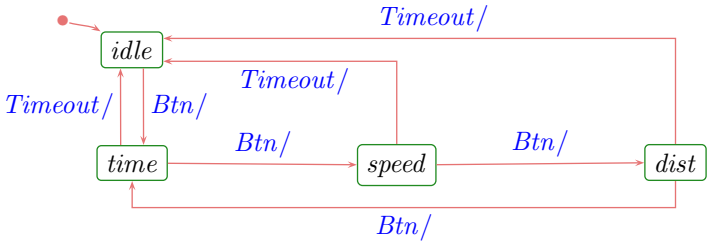


Figure 4: State machine model of the behaviour of a simple bicycle computer for Exercise 4.

Exercise 4 (5/20 Points)

From Exercise 4 of Exercise Sheet 4 you have a Rhapsody model of the state machine for a simple speedometer. For the next version of the speedometer, the state machine should be extended since requirements changed as follows:

1.) from state *idle*, signal *Btn* should move to the non-idle state (time, speed, or dist) we assumed before; use *time* is there was none before,

2.) the diagram should use only one transition for *Timeout*, not three,

3.) in each non-idle state (and only in the non-idle states), a display illumination should be available; it should be switched on with signal *BtnLP* (button pressed for a long time) and off after a shorter timeout indicated by signal *TimeoutShort*.

(i) Give a corresponding hierarchical state machine as Rhapsody model. (2)

(ii) For each requirement, demonstrate that your design is not "completely broken" by providing at least one recorded sequence diagram which demonstrates correct functioning of your design wrt. the requirement. (2)

(iii) Which aspect(s) of the state machine behaviour are not precisely fixed by the requirements 1.) to 3.)?

If you prefer to resolve it on your own, explain why this is an appropriate approach in this case, otherwise, ask the customer (your tutor) and point out how you considered the answer in your design. (1)

Exercise 5 (2/20 Points)

In the lecture, we discussed deep and shallow history connectors. Do they add expressive power to UML state machines or can they be viewed as abbreviations for something. In other words: can we realise history-connector like behaviour in core state machines? (2)

Exercise 6 (10 Bonus)

Note: this kind of exercise appears for the first time with the UML lecture. So there is a higher risk that things do not immediately work as expected, but correspondingly a higher gain (higher number of points).
Rhapsody comes with a component called *automatic test case generation*.
It takes a UML model and generates a suite of test cases, where each test case is a set of input stimuli (can be viewed as sequence diagram), such that executing all test cases on the model, one after the other, right after a fresh initialisation of the system, visits all states in the model at least once and uses each transition at least once.

(i) Generate test cases for your speedometer model. Submit your Rhapsody model including generated test cases (see below).

Discuss what you expected to get vs. what you did get. (8)

(ii) Do you have any idea what these test cases could be useful for? (2)

Hint:

- *The relevant manual is 'ATG User Guide' in the list of books (ATG_User_Guide.pdf).*
- *The task should be solvable after considering at most pages 36–57 (yet nobody stops you from reading the whole manual).*

  Most default settings apply, yet you need to explicitly mark your speedometer class and the signals which should be considered during test case generation.
- *After generating test cases, use Tools → Export to... → Rhapsody to export them to the Rhapsody model repository (pages 80ff in the manual).*

  The sequence diagram(s) end up in Package 'ATG' (navigate to leafs of the tree).