Important note: *SVT is a course that provides a formal base for the used concepts, algorithms and tools. This allows you to be* concise, short *and* exact *in your answers. Answers that are unnecessarily long, verbose, or not to-the-point can be subject to point-reduction. If the formulation of a question contains "Argue your answer" and you provide an answer without an argument, then* no *points will be awarded.*

You may answer the questions in English or in Dutch, and you can answer the questions in any order.

For this exam you may use clean *course materials, notably the workbook and the book "Model Checking. E.M. Clarke, O. Grumberg and D. Peled.". The paper by Tretmans is attached to the exam.*

You can get up to 100 points. The final score of the exam is the number of points divided by 10. You are allowed to use clean *versions of the workbook and the book "Model Checking" by Clarke et al. A version of the paper of Tretmans has been added as appendix to this exam.*

Question 1: Kripke structures **(10 points)**

Consider the following program code. In the initial state, variable x has some unknown value from 1 up to and including 4 and variable y has value 4.

```
while (x <= y && y > 0)
do
     y := y - x
end while
x := 0
```

Provide a Kripke structure modelling this program code.

Question 2: Timed automata **(20 points)**

Figure 1 shows a timed automaton in UPPAAL's notation.

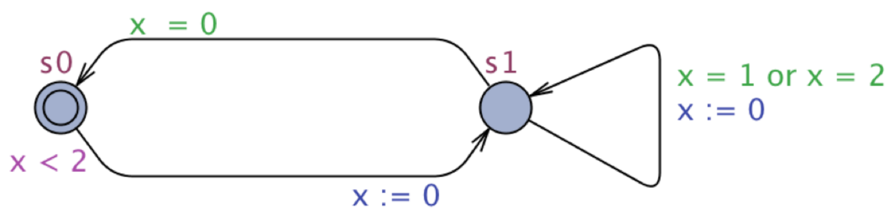


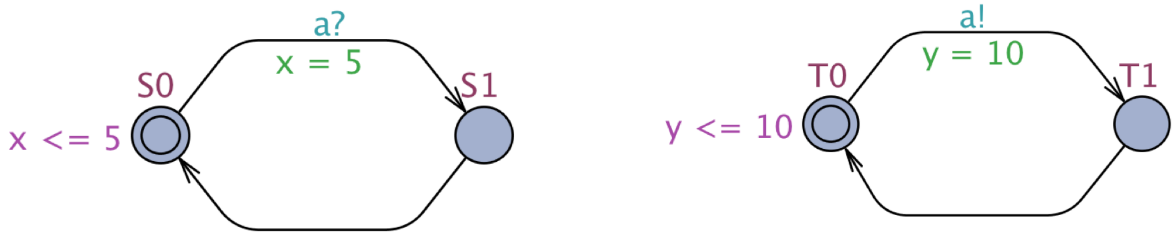
FIGURE 1 Timed automaton (UPPAAL notation). Note that we use := for an assignment in an update, and the = sign for equality in guards.

(5 points) (a) Draw the corresponding region automaton.

(3 points) (b) Does the UPPAAL automaton have a deadlock? If yes, then provide a trace leading to that deadlock. If not, then argue why not. In both cases, use the region automata in your argument.

(2 points) (c) Does the UPPAAL automaton exhibit Zeno behavior? Argue your answer. Use the region automaton in your argument.

(10 points) (d) Figure 2 shows two UPPAAL automata. Provide the parallel composition of these two automata.



(A) (B)

FIGURE 2 UPPAAL automata.

Question 3: Temporal logics **(15 points)**

(2 points) (a) Recall the timed automaton from Figure 1. Does the following CTL formula hold? Argue your answer. You may use the region automaton you constructed in Question 2 (a) in your answer.

$$\mathbf{AG}(s_1 \to \mathbf{AF}(s_0))$$

(3 points) (b) Consider the Kripke structure you constructed in Question 1. For each of the following two CTL formulae, state whether the formula holds for your Kripke structure. Argue your answer. Note that $\leadsto$ is UPPAAL's liveness operator and $\to$ is normal logical implication.

- $x \in \{0,2,4\} \leadsto (x = 0 \land y = 0)$
- $\mathbf{EG}(x \in \{0,2,4\} \to y \in \{0,2,4\})$

(10 points) (c) Provide a Kripke structure that demonstrates the difference between the CTL formulae $\mathbf{AF}(p)$ and $\mathbf{AG}\ \mathbf{AF}(p)$. Thus, the Kripke structure must be true for *exactly* one of these CTL formulae.

Question 4: Explicit model checking **(10 points)**
Figure 3 shows a Kripke structure. Execute the algorithm for explicit state model checking step-by-step for the following CTL formulae. This entails showing the steps performed, and providing the output of the algorithm.

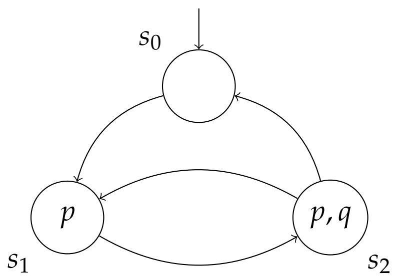


FIGURE 3 Kripke structure

(5 points) (a) $s_0 \models \mathbf{EF}\ \mathbf{AG}(p)$

(5 points) (b) $s_0 \models \mathbf{AF}\ \mathbf{EG}\ p$

Question 5: Symbolic model checking **(20 points)**
Figure 4 shows a Kripke structure.

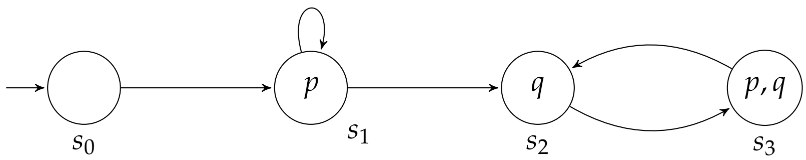


FIGURE 4 Kripke structure

(10 points) (a) Provide a BDD that symbolically represents the transition relation of the Kripke structure. Use variable ordering p', q, p, q'.

For each of the following formulas, apply the algorithm for symbolic model checking to the Kripke structure from Figure 4, by showing the evolution of set Z for each step of the algorithm. You do not have to give the actual BDDs, but you can give the sets of states represented by the BDDs.

(5 points) (b) $\mathbf{EG}(p)$

(5 points) (c) $\mathbf{AF}(p \wedge q)$

Continue on the next page...

Question 6: Labelled transition systems **(10 points)**
Figure 5 shows two IO automata: an implementation i and a specification s.

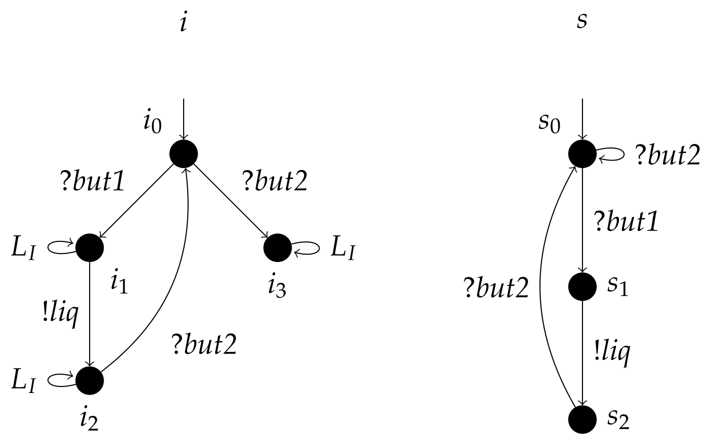


FIGURE 5 IO automata. Input alphabet $L_I = \{?\text{but1}, ?\text{but2}\}$. Output alphabet $L_U = \{!\text{liq}\}$

(2 points) (a) Is the implementation, i, deterministic? Argue your answer.

(2 points) (b) Is the specification, s, input-enabled? Argue your answer.

(2 points) (c) Consider the implementation i and specification s. Which states in these two LTSs are quiescent?

(2 points) (d) For each of the following predicates, state whether it is *true* or *false*.
(i) i_0 **after** $?but1$ **refuses** $\{!liq\}$
(ii) i_0 **after** $?but1$ **refuses** $\{!liq, \delta\}$
(iii) i_0 **after** $?but1 \cdot !liq$ **refuses** $\{!liq, \delta\}$
(iv) s_0 **after** $?but2 \cdot ?but1$ **refuses** $\{\delta\}$

(2 points) (e) Give a suspension trace of the implementation that is not a trace. If no such suspension trace exists, argue why not.

Question 7: IOCO **(5 points)**
Consider implementation i and specification s from Figure 5. Does the implementation conform to the specification, according to the **ioco** definition? Argue your answer.

Question 8: Test case generation & execution **(10 points)**
Consider implementation i and specification s from Figure 5.

(5 points) (a) When the specification is used for model-based testing, test-cases are generated. Give an example of a test case, that includes at least the following:
pass, **fail**, θ, !, and ?

(2 points) (b) Give a test run of your test case with implementation i.

(3 points) (c) Argue that your test case is sound. What is the smallest change you can think of, that makes the test case unsound?