Formal Methods

First Mid-term Exam

1.	2.	3.	4.	5.	6.	Σ
10 points	6 points	8 points	6 points	12 points	8 points	50 points

Name: ______________________________

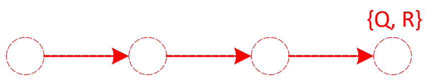
NEPTUN code: ______________________________

1. Theoretical questions (10 points)

1.1. For each of the following statements indicate (with an X) whether it is true, false or not decidable. **3 points**

Statement	True	False	Not decidable
Only one atomic proposition can be assigned to each state in Kripke structure models (KS).		X	
In bounded model checking of invariant properties, we need to increase the bound (and continue the algorithm) only if no counterexample was found with the current bound.	X		
It is possible that a node in an ROBDD has only one outgoing edge.		X	

1.2. Give a sequence of labeled states for which the property **F (P U (Q Λ R)** holds but the property **Q**, the property **X (P ∨ R)** and the property **XX Q** do not hold, using *as few states as possible*! **3 points**



1.3. Give an *example* for a temporal logic expression that is syntactically valid in PLTL (assuming the implicit path quantifier "A") but invalid in CTL. *Explain* why it is invalid in CTL! **2 points**

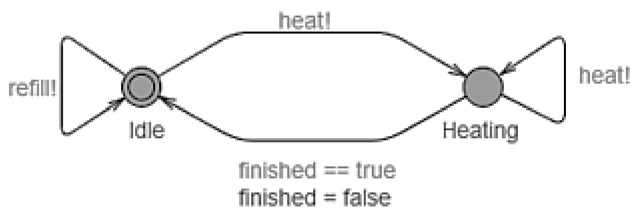
e.g.: A(**XX** *p*) or A(**X** *p* ∨ **F** *q*), because the path expressions may not be combined in CTL.

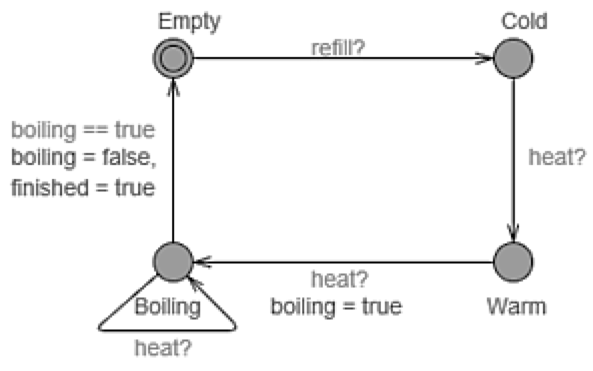
1.4. Describe the behavior of *urgent* states in timed automata (of UPPAAL)! **2 points**

The automaton cannot wait in an urgent location unless nothing else can happen in the system. It has to leave the urgent state as soon as possible.

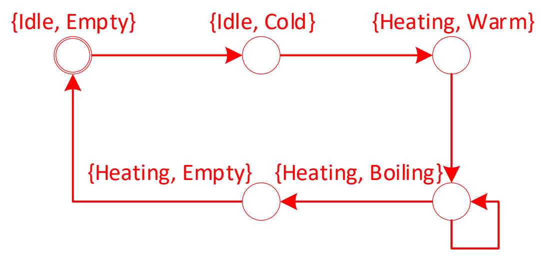
2. Modeling formalisms (6 points) **Please provide the solution on a new sheet!**

The following figures show two timed automata (modeled in UPPAAL) that describe the states of the controller of a heater (*Idle*, or *Heating*), and the states of the water in a water tank (*Empty*, *Cold*, *Warm* or *Boiling*. The automata use two logical variables (*bool boiling*, *finished*), and two channels (*chan refill*, *heat*). Logical variables are initially false. Note, that guards use "==" whereas assignments use "=".



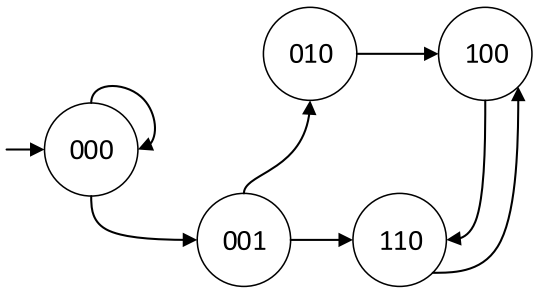


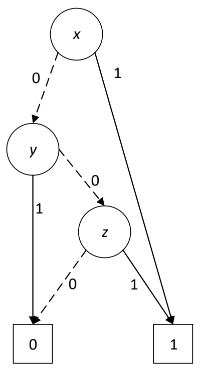
2.1. Construct the Kripke structure corresponding to the *whole system*, i.e., reachable combinations of the states of the controller and states of the water tank, including the transitions! Label each combined state with the names of the states that it represents (you can use the initial letters of the states)! **6 points**



3. Binary Decision Diagrams (8 points) **Please provide the solution on a new sheet!**

A Kripke structure is given in the left side of the figure, where the states are encoded in three bits using the variables x, y, z (for example 010 corresponds to x=0, y=1, z=0).



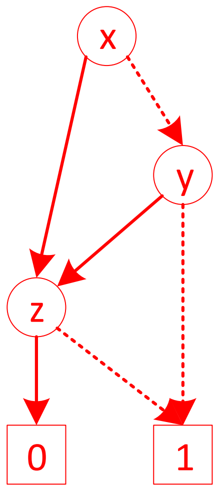


3.1. Give the characteristic function for the *initial state* of the Kripke structure!
Give the characteristic function for the *transitions outgoing from the initial state*! **2 points**

$$C_{000} = \neg x \land \neg y \land \neg z$$
$$C_{000\to 000} = (\neg x \land \neg y \land \neg z) \land (\neg x' \land \neg y' \land \neg z')$$
$$C_{000\to 001} = (\neg x \land \neg y \land \neg z) \land (\neg x' \land \neg y' \land \quad z')$$

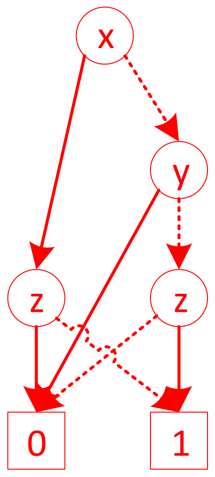
3.2. Draw the ROBDD representing the *states* of the Kripke structure! Use the following order for the variables: *x*, *y*, *z*!

3 points



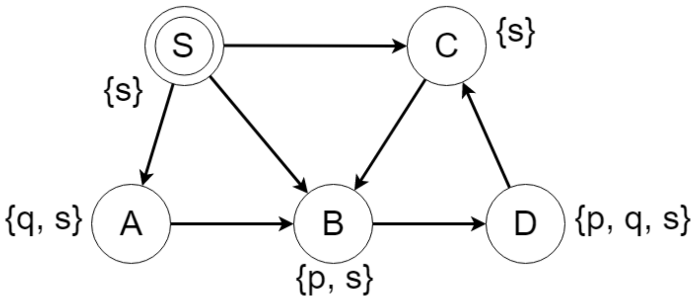
3.3. Give the ROBDD corresponding to the *intersection* of the states of the Kripke structure (constructed in the previous task) and the states encoded by the ROBDD in the right-hand side of the figure, using ROBDD operations! The variable order should remain *x*, *y*, *z*.

3 points



4. CTL model checking (6 points) **Please provide the solution on a new sheet!**

Consider the following Kripke structure:



4.1. Check if the following CTL expression holds from the initial state using the *iterative labeling algorithm* presented in the lectures: **A ((s ∨ p) U (AX q))**.
For each iteration give the expression that is currently used for labeling and enumerate the states that are labeled!

6 points

1. iteration: (labeling with **s ∨ p**)
   S, A, B, C, D
2. iteration: (labeling with **AX q**)
   B
3. iteration: (labeling with **A ((s ∨ p) U (AX q))**)
   B (because of **AX q**)
   A, C (first iteration backward, labeled with **p ∨ q**, all successors already labeled)
   S, D (second iteration backward, labeled with **p ∨ q**, all successors already labeled)
   (S may be labeled along with A and C if it is considered last.)

5. LTL requirement formalization (12 points) **Please provide the solution on a new sheet!**

On each day we record the status of solar activity and the command that we give to a satellite to protect itself. Solar activity can be *calm*, *weak* or *strong*. The satellite can have *closed* solar panels and/or it can turn *away* from the solar activity.
Use LTL expressions to formalize the following three requirements, which must apply to the behavior of the system *in every moment*!

5.1. If solar activity is *weak* or *strong*, the satellite is turned *away* until solar activity becomes *calm*.

2 points

G((*weak* ∨ *strong*) ⇒ *away* **U** *calm*))

5.2. If solar activity is *strong* the satellite eventually *closes* its panels or turns *away*.

2 points

G(*strong* ⇒ **F**(*closed* ∨ *away*))

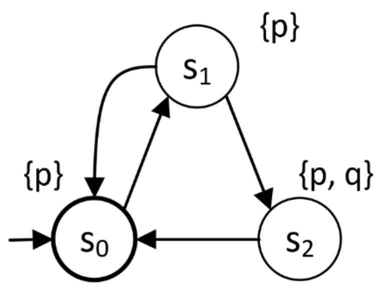
5.3. If solar activity is *calm* for three consecutive days, the satellite does not *close* its panels on the second day and on the third day neither the panels are *closed* nor is the satellite turned *away*.

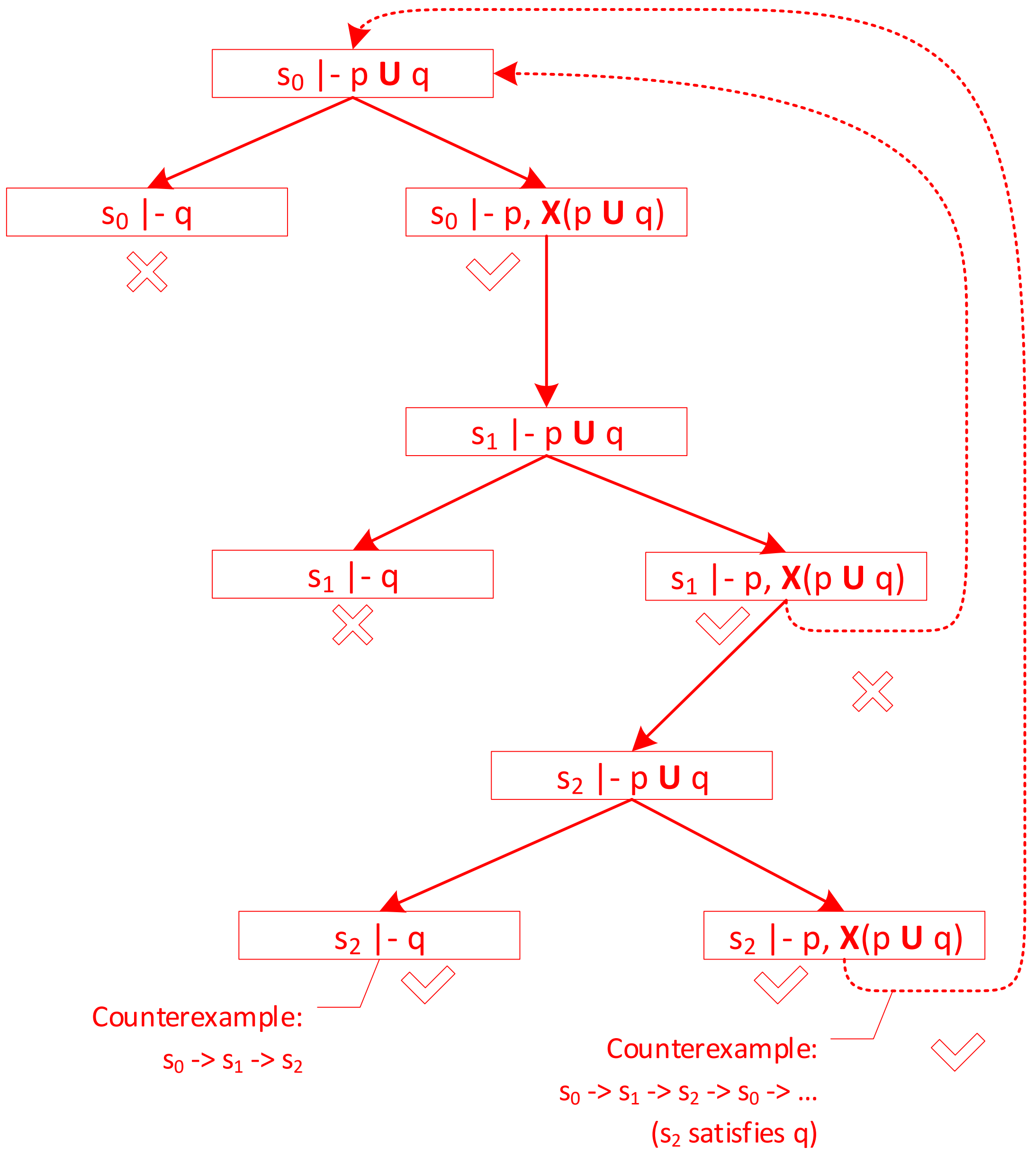
2 points

G((*calm* ∧ **X** *calm* ∧ **XX** *calm*) ⇒ (**X**(¬*closed*) ∧ **XX** (¬*closed* ∧ ¬*away*)))

5.4. Use the *tableau method* to check if the requirement ¬(p **U** q) holds for the Kripke structure below! *Explain and document* your solution! If the requirement does not hold, give a counterexample *based on the tableau*!

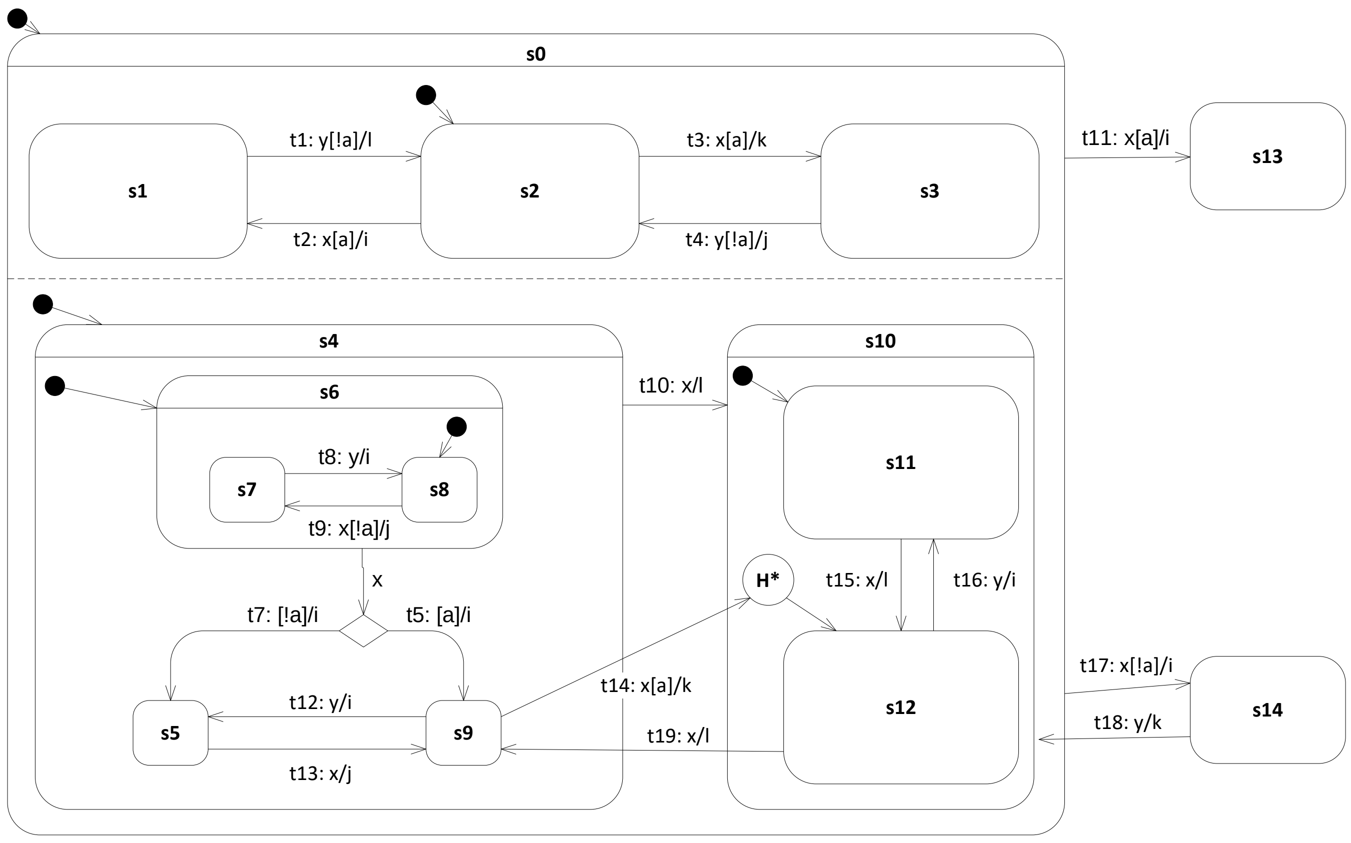
6 points





6. UML statecharts (8 points)

Consider the following statechart, in which for all states s_k there is also an entry action s_k*.entry* and an exit action s_k*.exit* that is not displayed in the figure! The expressions on the arrows (transitions) have the following form: *transition_name: trigger [guard] / action*.



The statechart starts from the default initial state, the value of the logical variable *"a"* is *"true"*. The incoming event is *"x"*.

6.1. Which transitions are *enabled*? | 1 point

t2, t3, t5, t10, t11

6.2. Which enabled transitions are *in conflict (cannot fire together)*? | 1 point

(t11, t2), (t11, t3), (t11, t5), (t11, t10), (t10, t5), (t2, t3)

6.3. What is the set of *fireable* transitions after resolving the *conflicts*? If there are multiple sets of fireable transitions, give *all* sets! | 1 point

{t2, t5}, {t3, t5}

6.4. What is (are) the *next stable* state configuration(s)? If there are more than one possible stable state configurations give *only one* of them! Give the actions and their order during firing the transition! Do not forget to include the entry and exit actions! | 3 points

For {t2, t5}: Next: {s0, s1, s4, s9} Actions: (s2.exit, i, s1.entry) || (s8.exit, s6.exit, i, s9.entry)

For {t3, t5}: Next: {s0, s3, s4, s9} Actions: (s2.exit, k, s3.entry) || (s8.exit, s6.exit, i, s9.entry)

6.5. The next incoming event is again *“x”*. Give the set of fireable transitions (after resolving conflicts) and the next stable configuration! If there are more than one fireable sets and next stable configurations, give *all* of them!

2 points

From {s0, s1, s4, s9}: Fireable: t14 Next: {s0, s1, s10, s12}

From {s0, s3, s4, s9}: Fireable: t14 Next: {s0, s3, s10, s12}