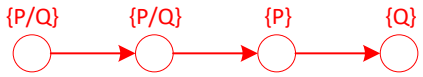
Formal Methods (VIMIM100)	Year 2017/2018, semester II						13. March 2018.
First Mid-term Exam, Group A	1.	2.	3.	4.	5.	6.	Σ
Name: _____							
NEPTUN code: _____	10 points	6 points	8 points	6 points	12 points	8 points	50 points

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
Each transition in a Labeled Transition System (LTS) can have only one action assigned.	X		
If bounded model checking finds a counterexample for an invariant property, it may be the case that it is not a counterexample if we use unbounded (full) model checking.		X	
Changing the order of variables in an ROBDD will always yield an ROBDD with the same size.		X	

1.2. Give a sequence of labeled states for which the properties **G (P ∨ Q)** and **XX (P U Q)** hold, but the property **XX Q** does 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 CTL* but invalid in CTL. *Explain* why it is invalid in CTL! **2 points**

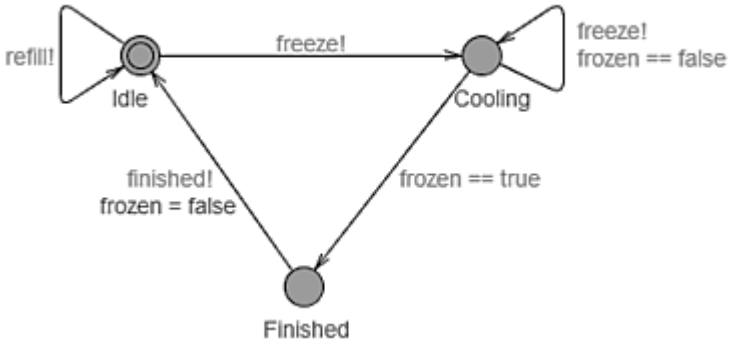
e.g.: **A**(**XX** *p*) or **E**(**X** *p* ∨ **F** *q*), because the path expressions may not be combined in CTL.

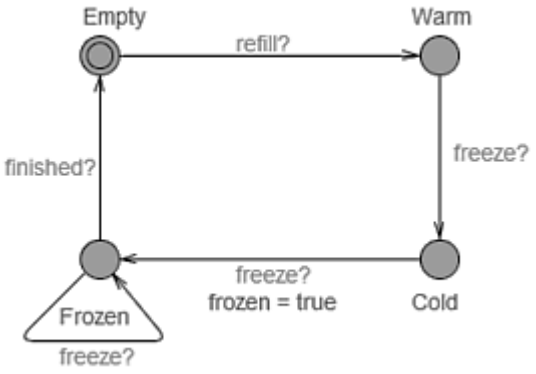
1.4. Describe the behavior of *committed* states in timed automata (of UPPAAL)! **2 points**

The incoming and outgoing transitions are executed as an atomic step, i.e., no other automaton can fire its transition while the *committed* location is active.

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

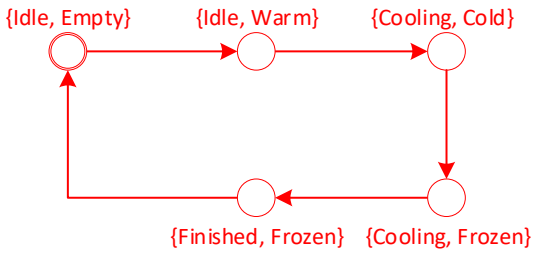
The following figures show two timed automata (modeled in UPPAAL) that describe the states of the controller of a cooler (*Idle*, *Cooling* or *Finished*) and the states of the water in a water tank (*Empty*, *Warm*, *Cold* or *Frozen*). The automata use a single logical variable (*bool frozen*), and three channels (*chan refill, freeze, finished*). The logical variable is initially false. Note that guards use "==" whereas assignments use "=".





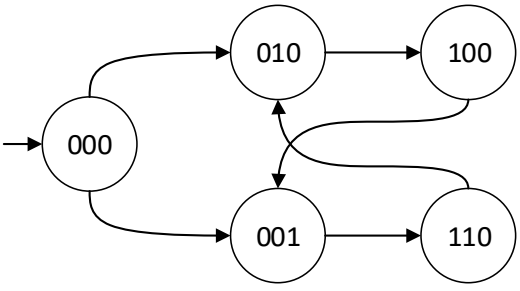
2.1. Construct the Kripke structure corresponding to the *whole system*, i.e., reachable combinations of the states of the controller and states of the water tank, including the transitions! Label each combined state with the names of the states that it represents (you can use the initial letters of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

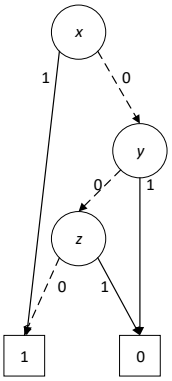
6 points



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).



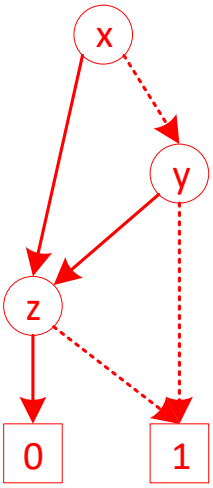


3.1. Give the characteristic function for the *initial state* of the Kripke structure!
Give the characteristic function for the *transitions outgoing from the initial state*!

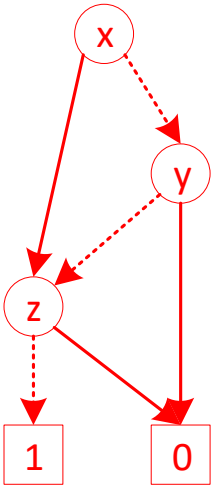
2 points

$$C_{000} = \neg x \wedge \neg y \wedge \neg z$$
$$C_{000\to001} = (\neg x \wedge \neg y \wedge \neg z) \wedge (\neg x' \wedge \neg y' \wedge z')$$
$$C_{000\to010} = (\neg x \wedge \neg y \wedge \neg z) \wedge (\neg x' \wedge y' \wedge \neg z')$$

3.2. Draw the ROBDD representing the *states* of the Kripke structure! Use the following order for the variables: *x*, *y*, *z*! 3 points

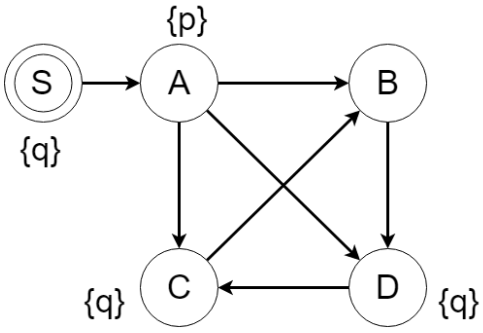


3.3. Give the ROBDD corresponding to the *intersection* of the states of the Kripke structure (constructed in the previous task) and the states encoded by the ROBDD in the right-hand side of the figure, using ROBDD operations! The variable order should remain *x*, *y*, *z*. 3 points



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: **E ((p ∨ q) 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 **p ∨ q**)
   S, A, C, D
2. iteration: (labeling with **AX q**)
   B, D
3. iteration: (labeling with **E ((p ∨ q) U (AX q))**)
   B, D (because of **AX q**)
   A, C (first iteration backward, both predecessors of B or D and labeled with **p ∨ q**)
   S (second iteration backward, predecessor of A and labeled with **p ∨ q**)

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

On each day we record the status of the weather and our equipment to protect ourselves. The weather can be *sunny*, *windy* or *rainy*. Our equipment can be an *umbrella* and/or a *coat*.
Use LTL expressions to formalize the following three requirements, which must apply to the behavior of the system *in every moment*!

5.1. If the weather is *rainy* we will eventually take an *umbrella* or a *coat* with us. — 2 points

$$\mathbf{G}(rainy \Rightarrow \mathbf{F}(umbrella \vee coat))$$

5.2. If the weather is *rainy* or *windy* we are taking our *coat* with us until the weather becomes *sunny*. — 2 points

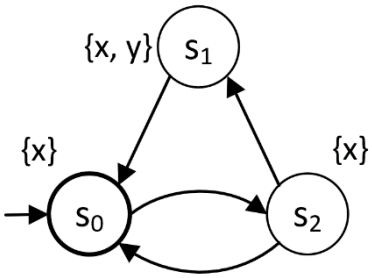
$$\mathbf{G}((rainy \vee windy) \Rightarrow (coat\ \mathbf{U}\ sunny))$$

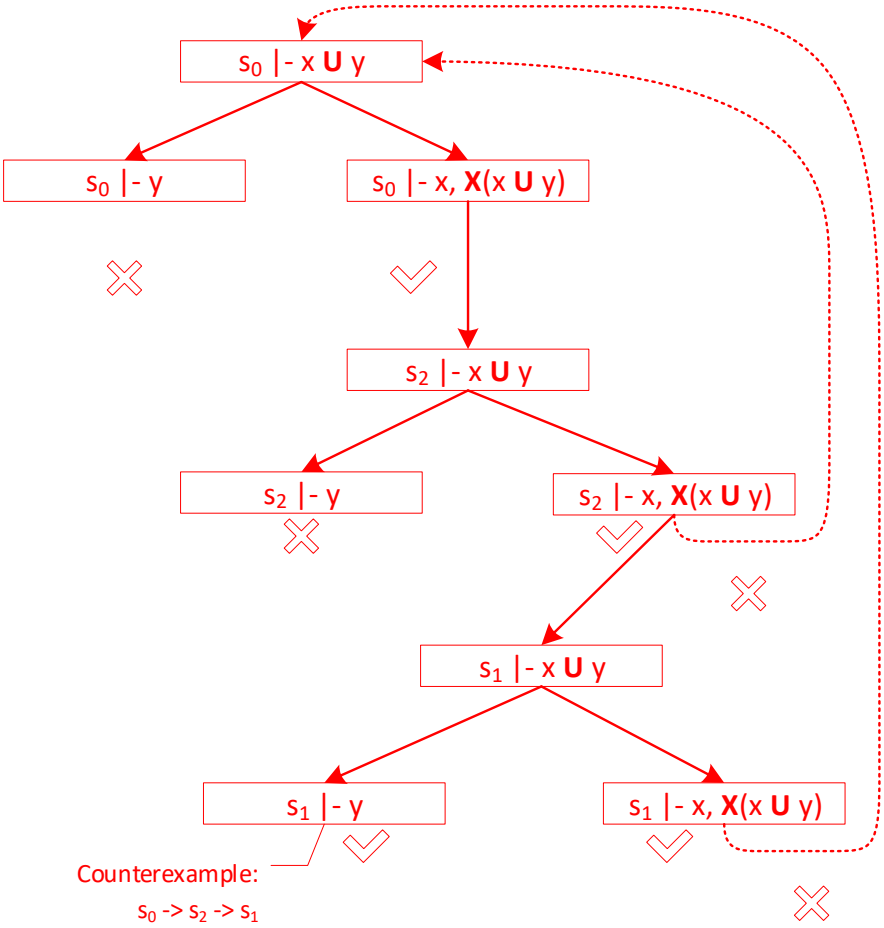
5.3. If the weather is *sunny* for three consecutive days, we are not bringing our *umbrella* on the second day and we will not take an *umbrella*, neither a *coat* on the third day. — 2 points

$$\mathbf{G}((sunny \wedge \mathbf{X}\ sunny \wedge \mathbf{XX}\ sunny) \Rightarrow (\mathbf{X}(\neg umbrella) \wedge \mathbf{XX}\ (\neg umbrella \wedge \neg coat)))$$

5.4. Use the *tableau method* to check if the requirement **¬(x U y)** 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

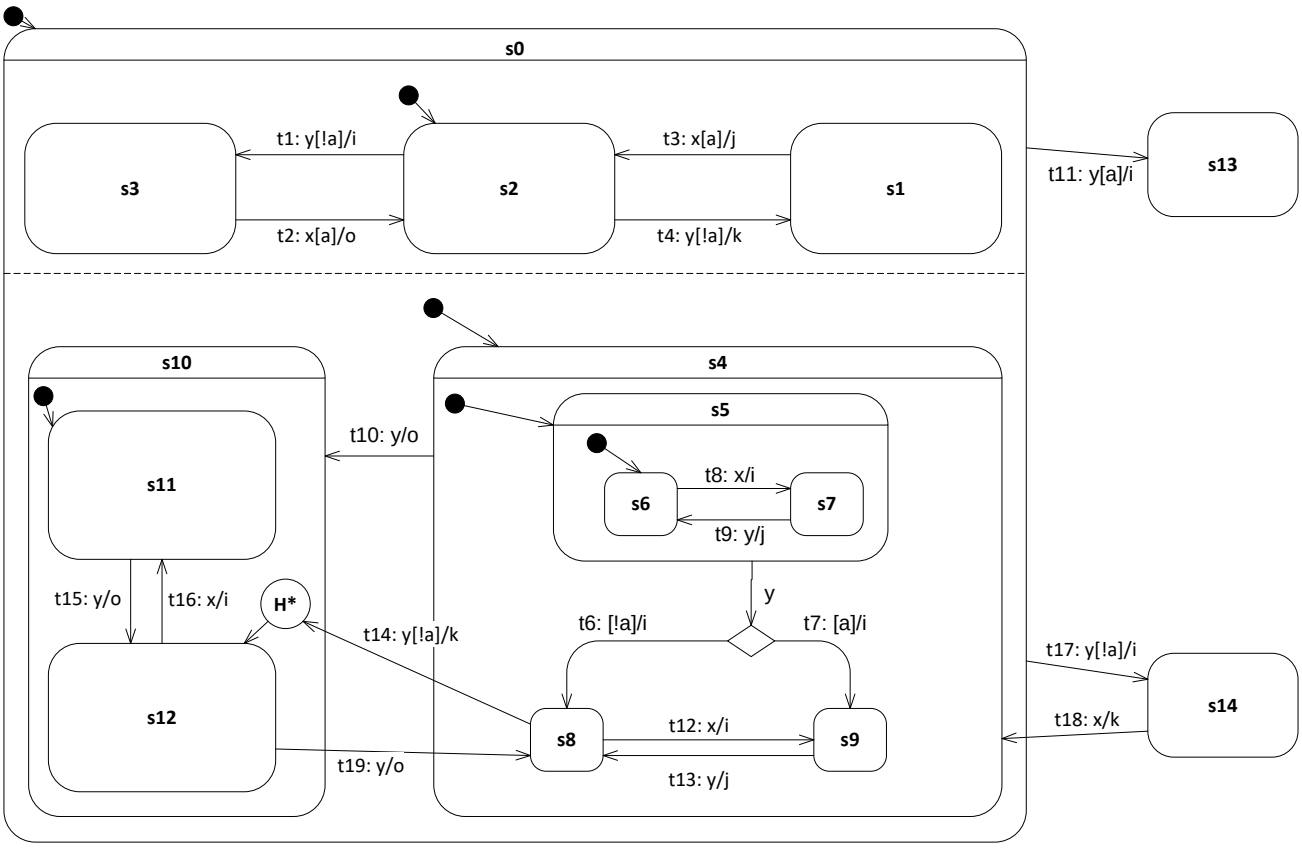
Negated requirement: **¬(¬(x U y)) = x U y**
Negated normal form: **x U y**


{x, y}
s1
{x}
{x}
s0
s2


s0 |- x U y
s0 |- y
s0 |- x, X(x U y)
s2 |- x U y
s2 |- y
s2 |- x, X(x U y)
s1 |- x U y
s1 |- y
s1 |- x, X(x U y)
Counterexample:
s0 -> s2 -> s1

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 *"false"*. The incoming event is *"y"*.

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

t1, t4, t6, t10, t17

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

(t17, t1), (t17, t4), (t17, t6), (t17, t10), (t10, t6), (t1, t4)

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

{t1, t6}, {t4, t6}

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 {t1, t6}: Next: {s0, s3, s4, s8} Actions: (s2.exit, i, s3.entry) || (s6.exit, s5.exit, i, s8.entry)

For {t4, t6}: Next: {s0, s1, s4, s8} Actions: (s2.exit, i, s1.entry) || (s6.exit, s5.exit, i, s8.entry)

6.5. The next incoming event is again *"y"*. 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, s3, s4, s8}: Fireable: t14 Next: {s0, s3, s10, s12}

From {s0, s1, s4, s8}: Fireable: t14 Next: {s0, s1, s10, s12}