Formal Methods (VIMIM100)	**2016/2017. year II. semester**					23. March 2017.
First Mid-term Exam	1.	2.	3.	4.	5.	Σ
Name: ______________________						
NEPTUN code: ______________________	12 points	14 points	8 points	8 points	8 points	50 points

1. Theoretical questions (12 points)

1.1. Give the *formal definition* of a *Kripke structure*! Describe the basic *difference* between a Kripke structure and a *labeled transition system* (LTS)! **3 points**

1.2. Describe the most important *restrictions* (constraints) in CTL compared to CTL*! Give an *example formula* that is a valid CTL* expression but not a valid CTL expression! **3 points**

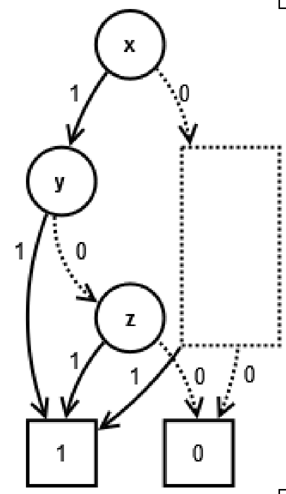
1.3. How can we reach a *contradicting branch* when using the *tableau decomposition* method for PLTL model checking and applying the decomposition rules for *1)* operator **X** *2)* and operator **U**? **3 points**

1.4. Describe the basic idea of *bounded model checking* (BMC) and describe with a *flow chart* how to use it in an *iterative* strategy to cover the state space up to a given bound! **3 points**

2. Binary Decision Diagrams (12 points) **Please provide the solution on a new sheet!**

2.1. Give the *reduced ordered binary decision diagram* (ROBDD) representation of the function f given by the truth table below! Use the variable order x, y, z in the ROBDD representation! — 3 points

x	y	z	$f(x, y, z)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

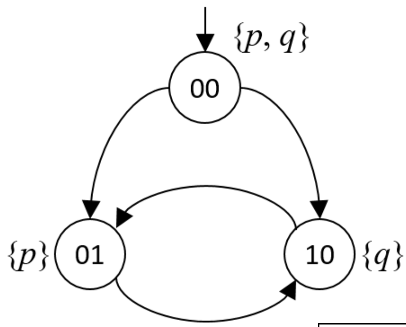


2.2. There is *exactly one* node missing from the dotted rectangle in the ROBDD above. (Note that this ROBDD is not related to the function f in the previous subtask.) Which variable(s) may belong to this missing node? *For each* possible variable, give the *algebraic form* of the function described by the ROBDD! — 4 points

2.3. Select *one* of the possible functions from the previous subtask (2.2) and denote this function by g. Compute the ROBDD representation of the function $f \vee g$! Perform the computation *directly using the ROBDD operations* on f and g (OR operation)! Use the same variable order as before (x, y, z)! — 7 points

3. Symbolic model checking (8 points) **Please provide the solution on a new sheet!**

The following Kripke structure is given including the bit vector encoding of the states:



3.1. Describe the characteristic function of *each state* (using variables x_1 and x_2)! — 1 point

3.2. Describe the characteristic function of the *set of states* labeled with p! — 1 point

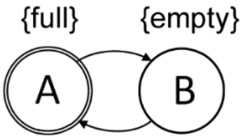
3.3. Describe the characteristic function of the *transition relation* (containing *all* transitions of the Kripke structure)! — 2 points

3.4. Run the semantics-based model checking procedure based on the *iterative labeling algorithm* to check if the CTL expression **A**((**EX** p) **U** $\neg q$) holds *for the initial state*! *For every step* of the iteration, give the labeling expression and enumerate the labeled states! — 4 points

4. LTL requirement formalization (8 points)

The behavior of a *beer* in a dormitory room is modeled with the Kripke structure on the right (the initial state is A). The behavior of *Steve* (a student in the dormitory) is defined by the following rules (the rules are in the form: `<condition> -> <state transition>`):

```
var steve: {sleeping, sober, drunk}
initialization:
      steve := sleeping
transition rules:
      steve = sober && beer = full -> steve := drunk
      steve = sober && beer = empty -> steve := sleeping
      steve = sleeping -> steve := sober
      steve = drunk -> steve := sleeping
```



Furthermore, we know that during the time Steve gets drunk, he drinks all the beer.

4.1. First, give the Kripke structure representing *Steve*. Note that in some cases the behavior of Steve depends on the state of the beer. In such cases assume that the beer can be in *any* state. Use the state labels {*sleeping*, *sober*, *drunk*} to describe Steve! Then, give the Kripke structure representing the *whole system* (i.e., Steve and the beer)! In this case, the states will be pairs. **2 points**

4.2. Use *LTL expressions* to formalize the following requirements, which must apply to the behavior of the system in every case! Use the labels introduced in the previous subtask! Note that the requirements may or may not hold for the actual system. **6 points**

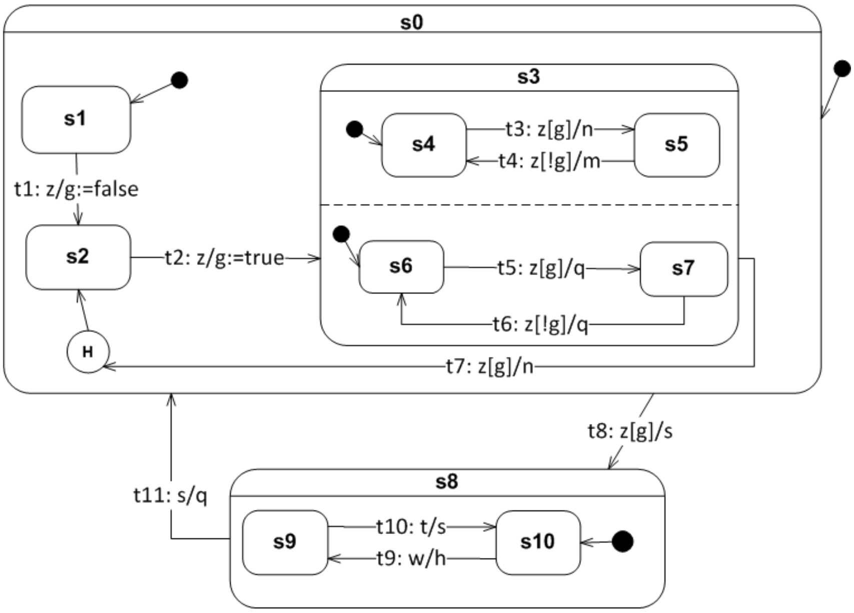
4.2-1. It is universally true that Steve will eventually be sober.

4.2-2. It is universally true that when Steve is drunk, he will be sleeping at the next step.

4.2-3. It is universally true that if Steve is sober, he will not get drunk until the beer is empty.

5. 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*. Guards are given as expressions, actions are given as letters (such as *n*) or by assignments (such as *g := true*).



The current state of the statechart is the following state configuration: (s0, s3, s4, s6) and the value of the logical variable *g* is *true*. The incoming event is *z*.

5.1. Which transitions are *enabled*? **1 point**

5.2. Which enabled transitions are *in conflict*? **1 point**

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

5.4. What is (are) the *next stable* state configuration(s)? If there are more than one possible stable state configurations, give *all* of them! Give the actions and their order during firing the transitions! Do not forget to include the entry and exit actions! **2 points**

5.5. Decide whether the following statements are true or false! Explain the answer (on a separate sheet)! Reachability is considered from the initial configuration with arbitrary incoming event(s). **3 points**

a) A configuration can be reached where t_8 is *fireable*.
b) The configuration (s0, s3, s4, s7) is reachable.
c) A configuration can be reached where the next configuration is *not deterministic* (there are more possibilities) even if the incoming event and the guard is known.