Formal Methods						
First Mid-term Exam	1.	2.	3.	4.	5.	Σ
Name: _____						
NEPTUN code: _____	12 points	12 points	8 points	10 points	8 points	50 points

1. Theoretical questions (12 points)

1.1. Argue if the following LTL equivalence is correct or not! Please explain your answer in detail! The sign "∨" represents the OR operator (from Boolean logic): **(F Stop) ∨ (F Start) = F (Stop ∨ Start)** — 3 points

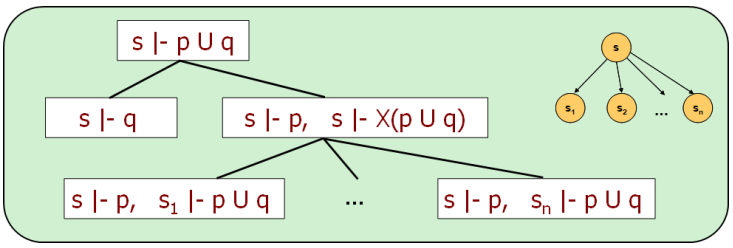
It is correct. There are 2 relevant cases:

- There is no state on the path labeled with Start or Stop: both expressions are false.
- There is a state along the path that is labeled with either Start or Stop: the left expression is true because one of the operands of the disjunction is true and the right expression is also true because the labeled state will satisfy F.

1.2. Argue if the formula **A(XX Stop ∨ F Start)** is a syntactically correct (valid) CTL and/or CTL* formula! Please explain your answers! If the expression is not syntactically correct, please explain the reason! — 3 points

It is not valid in CTL, because **XX Stop** or **A(…∨…)** is not allowed in CTL (only complex operands composed of a path quantifier and a temporal operand).
It is valid in CTL* because it is a state formula (because it starts with a path quantifier).

1.3. Describe the *tableau decomposition rule* for the PLTL operator **U** and describe the notation! How can we reach a contradicting branch when the decomposition rules for operator **U** are applied for model checking? Explain your answer! — 3 points



A branch is contradicting if 1) atomic propositions contradict each other or the labeling of the current state, 2) there is no next state and we have not seen a state labeled with *q*, and 3) there is a loop labeled with *p* but not with *q*.

1.4. Describe the transformation rules applied on a decision tree representation of a Boolean function to get a Reduced Ordered Binary Decision Diagram (ROBDD) representation of the same function! — 3 points

1) (Ordered) Order the test variables on every branch (in the same order).
2) (Decision Diagram) Merge identical sub-trees.
3) (Reduced) Discard nodes with both edges pointing to the same node.

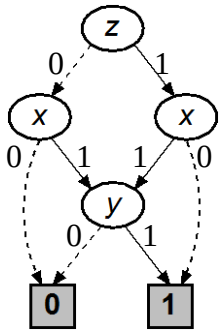
2. Binary Decision Diagrams (12 points)

Please provide the solution on a new sheet!

The definition of the binary functions:

$f := (x \wedge \mathbf{A}) \vee (\neg x \wedge \mathbf{B})$
$g := \neg (\neg y \wedge z) \vee (\neg y \wedge \neg z)$
$m := f \wedge g$

ROBDD representation of the function f:



2.1. Give the algebraic form of function *f* according to the ROBDD representation! **2 points**

$$(z \wedge \neg x) \vee (z \wedge x \wedge y) \vee (\neg z \wedge x \wedge y) = (z \wedge \neg x) \vee (x \wedge y)$$

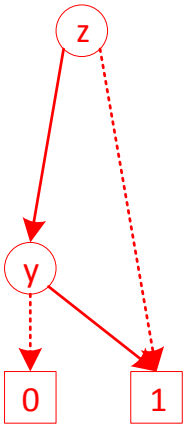
2.2. What variable(s) should be written in place of the placeholders **A** and **B** in the definition of function *f* to reflect the meaning of the ROBDD? **2 points**

Based on the previous solution: $\mathbf{A} = y; \mathbf{B} = z$

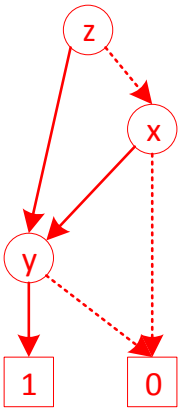
2.3. Give the Reduced Ordered Binary Decision Diagram (ROBDD) representation of the functions *g* and *m*! Use the following variable order in the ROBDD representation: *z*, *x*, *y*! Compute the ROBDD representation of function *m* by directly using the ROBDD operations on *f* and *g* (AND operation)! **8 points**

Construction of g:

$g = \neg(\neg y \wedge z) \vee (\neg y \wedge \neg z)$
$g_z = \neg(\neg y \wedge 1) \vee (\neg y \wedge \neg 1) = y$
$g_{zx} = y$
$g_{zxy} = 1$
$g_{zx\bar{y}} = 0$
$g_{z\bar{x}} = y$
$g_{zxy} = 1$
$g_{zx\bar{y}} = 0$
(g_z will be reduced)
$g_{\bar{z}} = \neg(\neg y \wedge 0) \vee (\neg y \wedge \neg 0) = 1$



m:



3. Symbolic Model Checking (8 points)

Please provide the solution on a new sheet!

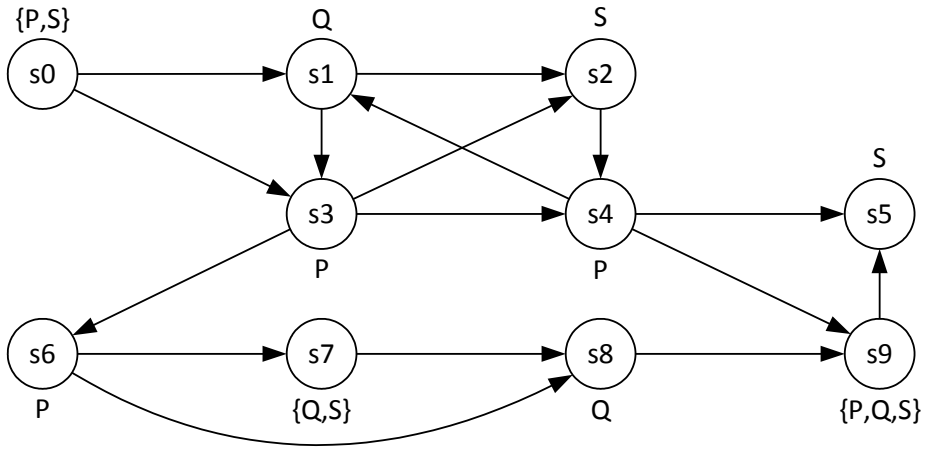
3.1. Describe the iterative labeling algorithm for the evaluation of the CTL formula **A(P U Q)**! How does the iterative labeling algorithm compute if a Kripke structure satisfies the CTL expression **A(P U Q)** (where **P** and **Q** are atomic propositions)? **4 points**

We use: $\mathbf{A}(p\ \mathbf{U}\ q) = q \vee (p \wedge \mathbf{AX}\ \mathbf{A}(p\ \mathbf{U}\ q))$
The iteration:

- States labeled *q* are the states where label $\mathbf{A}(p\ \mathbf{U}\ q)$ first appears
- We consider the predecessors of these states: If it is labeled with *p*, and all its successors are labeled with $\mathbf{A}(p\ \mathbf{U}\ q)$, we can add the label $\mathbf{A}(p\ \mathbf{U}\ q)$
- We terminate if no more label can be added.

3.2. Run the symbolic model checking procedure based on the iterative labeling algorithm evaluating the CTL expression $\neg S \wedge \mathbf{E}(P\ \mathbf{U}\ (Q \wedge S))$ on the given Kripke structure (**P**, **Q** and **S** are atomic propositions)! Give the assigned state labels in each step! Summarize the states and their corresponding labeling to decide which states satisfy the CTL expression!

4 points



$\underbrace{\underbrace{\neg S} \wedge \underbrace{\mathbf{E}(P\ \mathbf{U}\ (\underbrace{Q \wedge S}))}}$

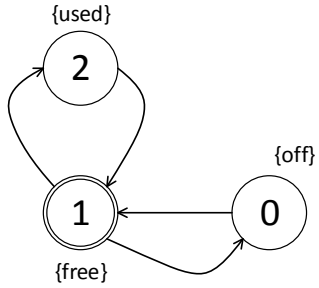
1. iteration: $Q \wedge S$
   s7, s9
2. iteration: $\mathbf{E}(P\ \mathbf{U}\ (Q \wedge S))$
   s7, s9, s6, s3, s0, s4
3. iteration: $\neg S$
   s1, s3, s4, s6, s8
4. iteration: $\neg S \wedge \mathbf{E}(P\ \mathbf{U}\ (Q \wedge S))$
   s3, s4, s6

States satisfying the CTL expression are s3, s4, s6 (no initial states was specified).

4. Modelling and Requirement Formalization (10 points)

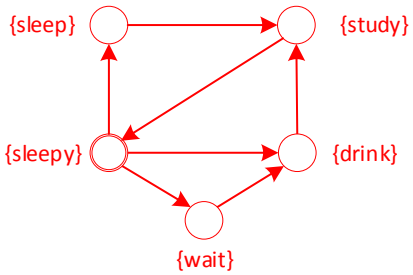
Coffee machine is one of the most important equipment in the life of a student. The operation of the coffee machine (*machine* for short in the example) is defined by the Kripke structure depicted on the left hand side figure (atomic propositions *used*, *free* and *off* are found in curly braces, they are used as state labels). The states and transitions of the student are defined by the rules on the right hand side (the rules are in the form: `<condition> -> <state transition>`).

Kripke structure: *machine*




```
var student: {sleepy, drink, wait, sleep, study}
initialization
    student := sleepy
transitions
    student = sleepy && machine = used -> student:= wait
    student = sleepy && machine = free -> student:= drink
    student = sleepy && machine = off -> student:= sleep
    student = drink -> student:= study
    student = wait && machine = free -> student:= drink
    student = sleep -> student:= study;
    student = study -> student:= sleepy
```

4.1. Give the Kripke structure of the behavior of the student! Use the state labels `{sleepy, drink, wait, sleep, study}`! — 2 points



4.2. Define a state encoding for the states and transitions of the Kripke structure of the coffee machine! Use two binary state variables: x and y for the state coding in a way that $x+2y$ gives the values of the states (values are written into the circles in the graphical representation, so `{used}=2`, `{free}=1` and `{off}=0`). By using this encoding, give the characteristic function for all the state transitions! — 2 points

used: $01 \rightarrow C_{used} = \neg x \wedge y$
free: $10 \rightarrow C_{used} = x \wedge \neg y$
off: $00 \rightarrow C_{used} = \neg x \wedge \neg y$

$$C_R = (\neg x \wedge \neg y \wedge x' \wedge \neg y') \vee (x \wedge \neg y \wedge \neg x' \wedge \neg y') \vee (x \wedge \neg y \wedge \neg x' \wedge y') \vee (\neg x \wedge y \wedge x' \wedge \neg y')$$

4.3. Use LTL expressions to formalize the following requirements, which apply to the behavior of the server in every case! Assume that we do not know the behavior of the system, only the set of atomic propositions (`{free, used, off}` and `{sleepy, drink, wait, sleep, study}`)! — 6 points

4.3-1. It is universally true that when the machine is being used, the student cannot drink.

$$\mathbf{G}(used \Rightarrow \neg drink)$$

4.3-2. It is universally true that when the machine is free, the student will drink at the next step.

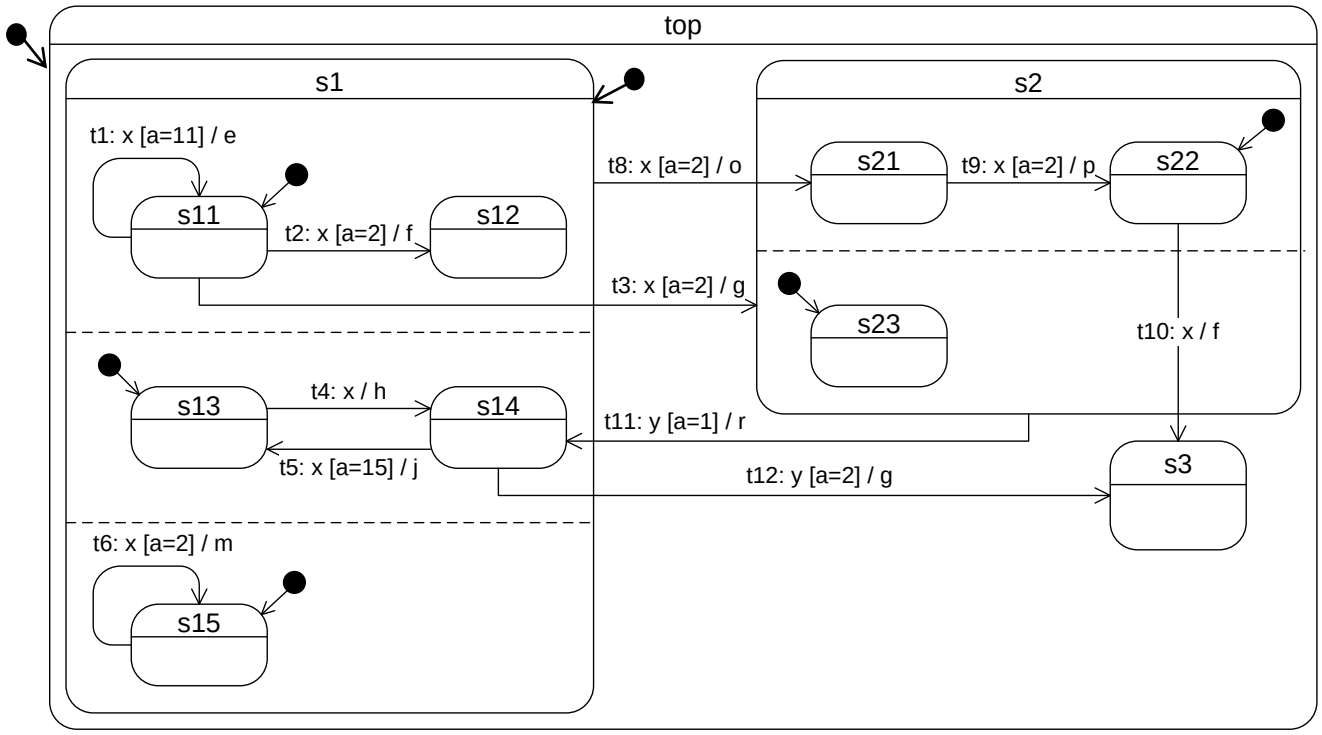
$$\mathbf{G}(free \Rightarrow \mathbf{X}\, drink)$$

4.3-3. It is universally true that the student does not have to wait infinitely long for drinking a coffee.

$$\mathbf{GF}(drink) \text{ or } \neg\mathbf{FG}(wait)$$

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 represented by letters (such as *f* or *h*). Assume that actions do not modify the value of variable *a*.



The current state of the statechart is the following state configuration: *{top, s1, s11, s14, s15}*. The value of variable *a* is 2. The incoming event is *x*.

5.1. Which transitions are enabled? — 1 point

t2, t3, t6, t8

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

(t2, t3), (t3, t6) (t2, t8), (t3, t8), (t6, t8)

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

{t2, t6} or {t3}

5.4. What is (are) the next stable state configuration(s)? If there are more than one possible stable state configuration, give all of them! — 2 points

For {t2, t6}: *{top, s1, s12, s14, s15}*
For {t3}: *{top, s2, s22, s23}*

5.5. The next incoming event is *x* again.
Give the set of fireable transitions, the actions being executed during firing, and the set of stable state configurations after the firing. If there were more stable state configurations after the former step, then give this information starting from all stable configurations! — 3 points

For {t2, t6}: fireable: t6 actions: s15.exit, m, s15.entry Next: *{top, s1, s12, s14, s15}*
For {t3}: fireable: t10 actions: {s22.exit, s23.exit}, s2.exit, f, s3.entry (here {}: in any order)
Next: *{top, s3}*