Formal Methods (VIMIMA07)	Year 2019/2020, Semester II					7. April 2020.
RME1 Repetition of the First Midterm Exam	1.	2.	3.	4.	5.	Total
Please provide each task on a separate page! Please indicate your name and Neptun code on each page!	7 points	5 points	5 points	6 points	12 points	35 points

1. Theoretical questions (7 points)

1.1. For each of the statements below, indicate whether they are *true*, *false*, or *not decidable*! — 3 points

A. In Kripke transition systems (KTS), transitions can have *guards* and a transition can only be taken if its guard is true.

B. The tableau method *can only* find counterexamples in the model that correspond to non-cyclical behavior.

C. The binary decision tree of a logical function *always contains more nodes* than its ROBDD representation.

1.2. Give a sequence of labeled states for which temporal properties **P U Q** and **X F(P ∧ Q)** hold, but the property **X Q** does not hold, using *as few states as possible*! — 3 points

1.3. Give an example temporal logic expression that is syntactically a *valid* CTL expression, but *not a valid* expression in PLTL extended with path quantifier A. — 1 point

Solution:

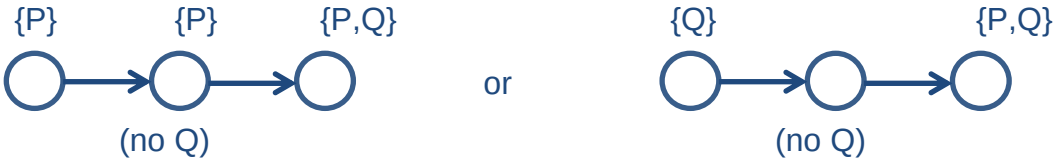
1.1

A: False. In KTS, transitions do not have guards.

B: False. The tableau method may find counterexamples that correspond to cyclical behavior (e.g., in case of the U operator).

C: False. It is possible that the binary decision tree and the ROBDD have the same number of nodes (e.g., in case of function f(x)=x, or function f(x,y) = x ↔ y).

1.2:



1.3:

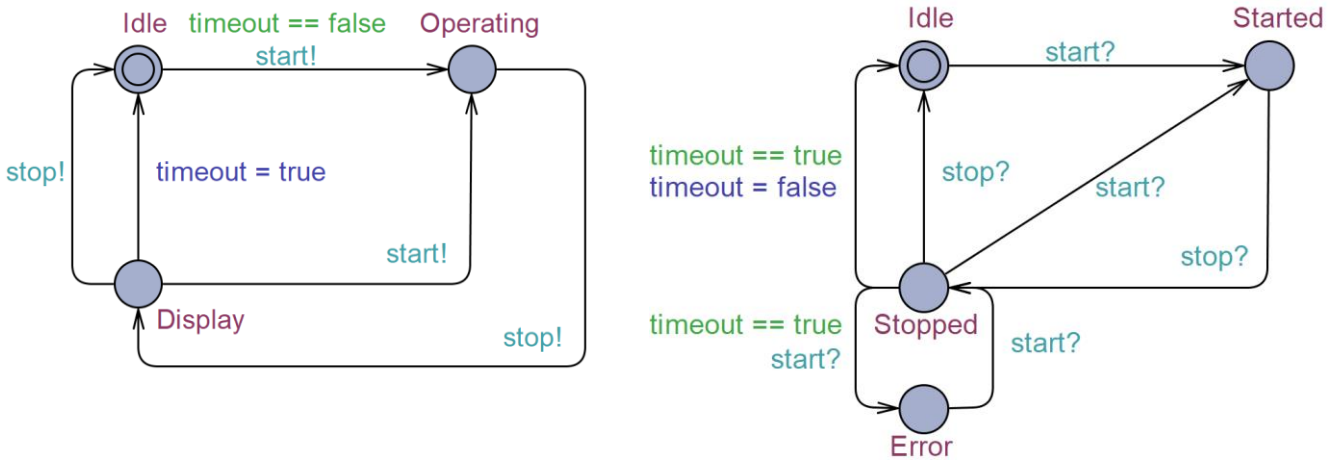
There are plenty of potential good examples.
E.g., the syntactically valid CTL expressions including one of the EX, EF, EG, EU temporal operators, or the syntactically valid CTL expressions that contain more than one CTL temporal operator.

2. Modeling (5 point)

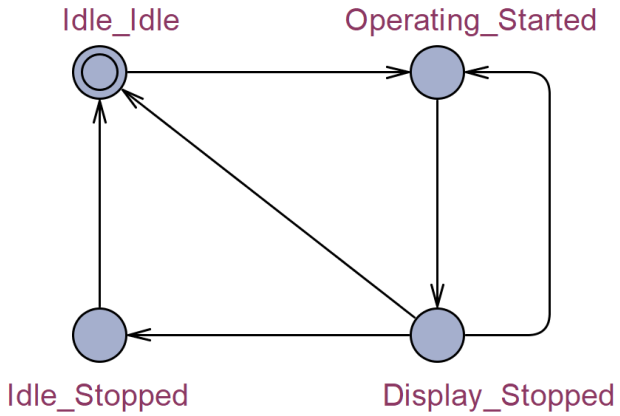
The following figures present two timed automata (modeled in UPPAAL) that describe the states of the controller of a digital stopwatch (*Idle*, *Operating*, *Display*), and the states of the related counter (*Idle*, *Started*, *Stopped*, *Error*). The automata use a single logical variable (*bool timeout*), and two channels (*chan start*, *chan stop*). 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 counter, and the *transitions* among the combined states. *Label* each combined state with the names of the states that it represents (you can use the initial letters of states for abbreviation)!

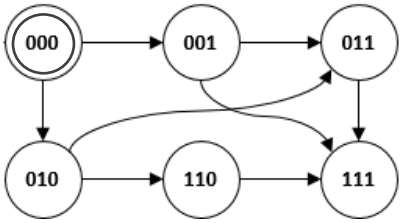
5 points



Solution:



3. Binary decision diagrams (5 points)



A Kripke structure is given on the right, where states are encoded with three bits using variables *x*, *y*, *z* in this order. (For example, the initial state encoded as 000 corresponds to *x*=0, *y*=0, *z*=0.)

3.1. Give the characteristic function for the *initial state* of the Kripke structure and the characteristic function for the *path* 000 → 001→ 111 starting from the initial state!	2 points
3.2. Draw the ROBDD representing the *set of states* of the Kripke structure using the variable order *x*, *y*, *z*!	3 points

Solution:

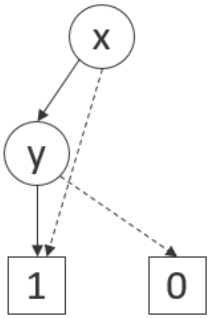
3.1:

$C_{000} = \neg x \wedge \neg \mathrm{y} \wedge \neg \mathrm{z}$

$C_{000\to001\to111} = (\neg x \wedge \neg \mathrm{y} \wedge \neg \mathrm{z}) \wedge (\neg x' \wedge \neg y' \wedge z') \wedge (x'' \wedge y'' \wedge z'')$

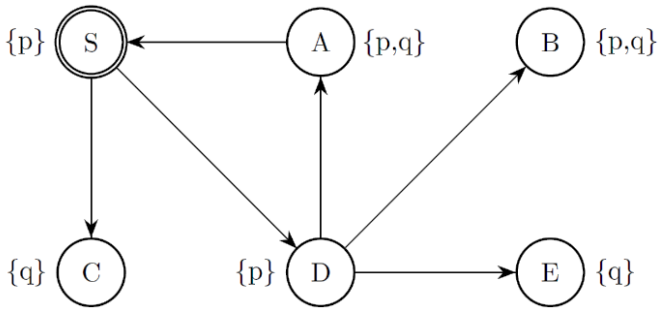
3.2:

The ROBDD is the following (it can be constructed by drawing first the binary decision tree of the function and then merging identical sub-trees and reducing redundant nodes):



4. CTL model checking (6 points)

Consider the Kripke structure on the right with initial state S and the given labeling.



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

6 points

Solution:

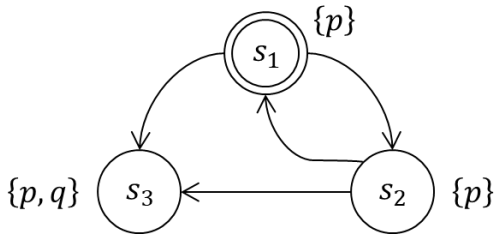
1. step: S, A, D states are labelled by **EX p** (there exists a direct successor state that is labeled by **p**).
2. step: A, B, C, E states are labelled by **A ((EX p) U q)** (these states are already labeled by **q**).
3. step: D state is labelled by **A ((EX p) U q)** (this state is labeled by **EX p** and all direct successor states are already labeled by **A ((EX p) U q)**).
4. step: S state is labelled by **A ((EX p) U q)** (this state is labeled by **EX p** and all direct successor states are already labeled by **A ((EX p) U q)**). End of the iteration.

The expression holds from the initial state because S is labeled by **A ((EX p) U q)**.

5. LTL requirement formalization and model checking (12 points)

In a chat application we can follow a conversation as an *observer*, as a *participant*, or we can be *away*. During the conversation we can *write* and *read* messages, and we can receive *alerts*. We record these facts in each time unit.
Formalize the following requirements using LTL operators and the given atomic propositions (denoted above by words in *italic*), which must *always* (continuously) apply to the behavior of the system!

5.1. If we are a *participant* in a conversation and we receive an *alert*, then we *write* a response in at most two time units.	2 points
5.2. We cannot *write* messages and cannot receive *alerts* until we follow the conversation as a *participant*.	2 points
5.3. An *observer* will eventually be *away* from the conversation.	2 points
5.4. Use the *tableau method* to check if the requirement **¬(p U q)** holds for the Kripke structure below (where the initial state is s_1)! If the requirement does not hold, give a counterexample *based on the tableau*!	6 points



Solution:

5.1: G ((participant ∧ alert) → (X write ∨ XX write))

5.2: G ((¬write ∧ ¬alert) U participant)

5.3: G (observer → F away)

5.4: The tableau belonging to **p U q** shall be constructed from state s_1. Counterexample on the satisfying branch: s_1, s_3 or s_1, s_2, s_3.