Formal Methods (VIMIMA07)	Year 2019/2020, Semester II					24. March 2019.
ME1A First Mid-term exam, Group A	1.	2.	3.	4.	5.	Total
Please start 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), states can be labeled with at most one atomic proposition and transitions can be labeled with at most one action.

B. Bounded model checking cannot be applied for a model which contains a loop (cycle).

C. The size of an ROBDD (corresponding to a logical function) is always independent from the variable ordering in the ROBDD.

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

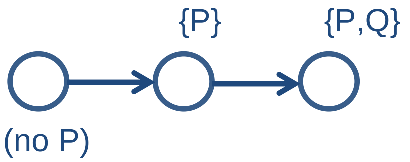
1.3. Give an example temporal logic expression that is syntactically *not a valid* CTL expression, but it is a *valid* CTL* expression. [1 point]

Solution:

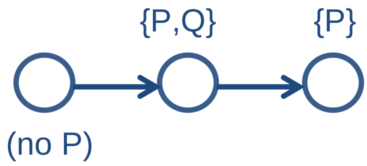
1.1

A: False. In case of KTS, transitions can be labeled with more than one action.
B: False. Bounded model checking can handle loops in models, as it considers loop-free *paths*.
C: False. The size of the ROBDD may depend on the variable ordering.

1.2:



or



1.3:

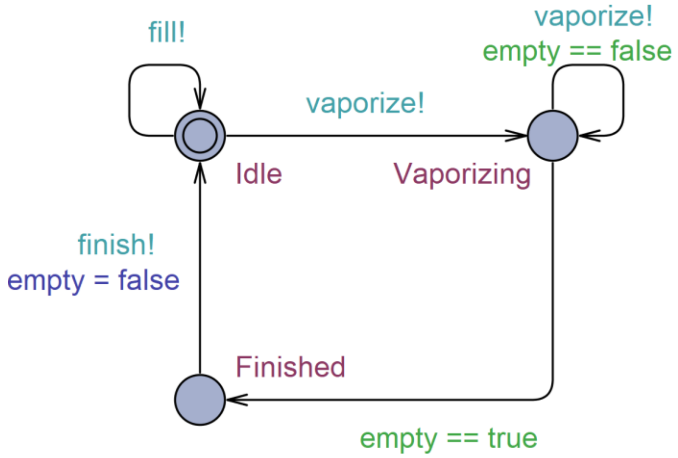
There are plenty of potential good examples.
E.g., in CTL, path formulas cannot be directly nested (these shall be directly preceded by path quantifiers E or A), while in CTL* these can be directly nested.

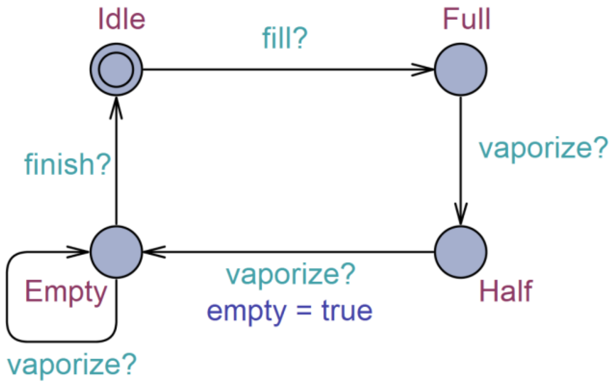
2. Modeling (5 points)

The following figures present two timed automata (modeled in UPPAAL) that describe the states of the controller of a vaporizer (*Idle*, *Vaporizing*, *Finished*), and the states of the vaporizer itself (*Idle*, *Full*, *Half*, *Empty*). The automata use a single logical variable (*bool empty*) and three channels (*chan fill*, *chan vaporize*, *chan finish*). The logical variable is initially false. Note that guards use “= =” whereas assignments use “=”.

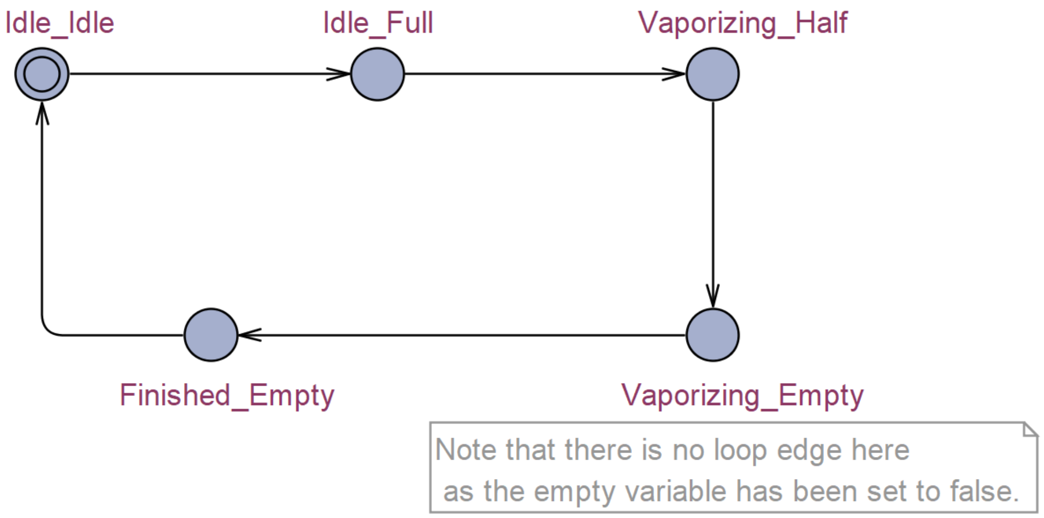
2.1. Construct the Kripke structure corresponding to the *whole system*, i.e., reachable *combinations of the states* of the controller and states of the vaporizer, and the *transitions* among the combined states. *Label* each combined state with the names of the states that it represents (you can use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

5 points

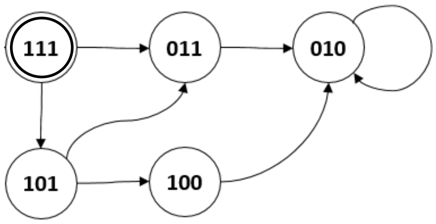




Solution:



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 111 corresponds to *x*=1, *y*=1, *z*=1.)

3.1. Give the characteristic function for the *initial state* of the Kripke structure and the characteristic function for the *path* 111 → 101→ 011 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:

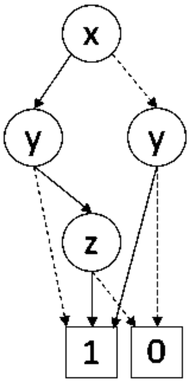
3.1:

$C_{111} = x \wedge \mathrm{y} \wedge \mathrm{z}$

$C_{111\to101\to011} = (x \wedge \mathrm{y} \wedge \mathrm{z}) \wedge (x' \wedge \neg y' \wedge z') \wedge (\neg x'' \wedge y'' \wedge z'')$

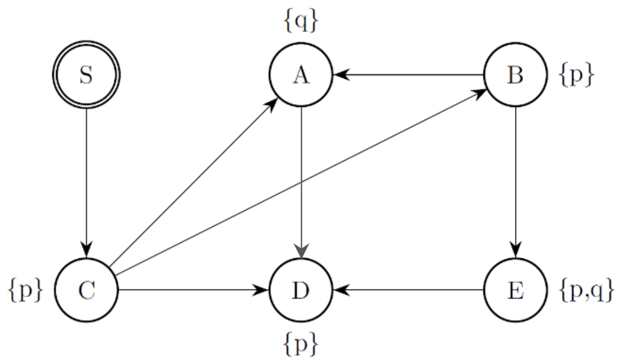
3.2:

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 ((¬p) U (EX 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:

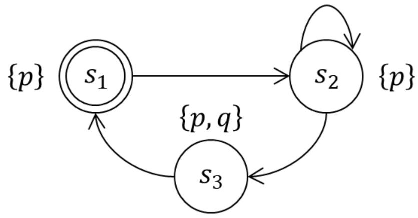
1. step: S, A states are labeled by **¬p**.
2. step: B, C states are labeled by **EX q** (there exists a direct successor state that is labeled by **q**).
3. step: B, C states are labeled by **A ((¬p) U (EX q))** (these states are already labeled by **EX q**).
4. step: S state is labeled by **A ((¬p)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

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

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

A video conference application supports *QVGA*, *VGA* and *SVGA* resolutions (the resolutions increase in this order). The network load can be *low* or *high* and it may happen that the video is *lagging*. We record these facts in reach minute.
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 the network load is *low* and the video is not *lagging*, then in the next minute we switch from *QVGA* to *VGA* resolution, and one minute afterwards we switch to *SVGA.*	2 points
5.2.	The video is *lagging* until the network load stays *high.*	2 points
5.3.	If the video is *lagging* on *SVGA* or *VGA* resolutions, we eventually switch to a lower resolution (from *SVGA* to *VGA* or *QVGA*, and from *VGA* to *QVGA*, respectively).	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 ((low ∧ ¬lagging ∧ QVGA) → (X VGA ∧ XX SVGA))

5.2: G (lagging U (¬high))

5.3: G (((lagging ∧ SVGA) → F (VGA ∨ QVGA)) ∧ ((lagging ∧ VGA) → F QVGA))

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