Formal Methods (VIMIMA07)	Year 2019/2020, Semester II					24. March 2019.
ME1B First Mid-term exam, Group B	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 Labeled transition systems (LTS), a transition can be labeled with an arbitrary number of action labels.

B. We can never find a counterexample for invariant properties with bounded model checking because that requires an infinite number of iterations.

C. A logical function must always be transformed into negation normal form (NNF) before creating its ROBDD form.

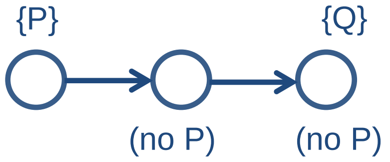
1.2. Give a sequence of labeled states for which temporal properties **F P** and **XX(P U Q)** hold, but the property **X(P ∨ X P)** does not hold, using *as few states as possible*! **3 points**

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

1.1

A: False. In case of LTS, a transition can be labeled with only one action.
B: False. Bounded model checking can be used to find counterexample for invariant properties.
C: False. The transformation into negation normal form is not necessary.

1.2:



1.3:

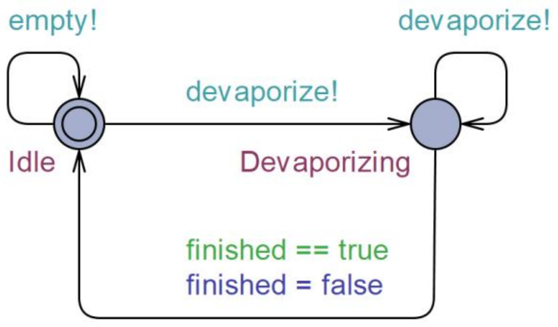
There are plenty of potential good examples.
E.g., in CTL, path formulas cannot be directly nested (these shall be directly preceded by path quantifiers E or A), while in CTL* these can be directly nested.

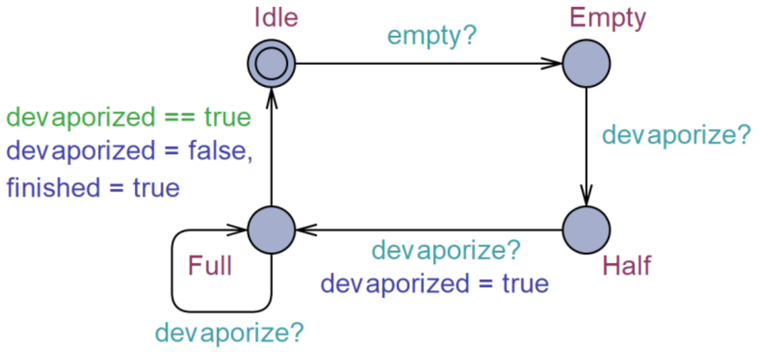
2. Modeling (5 points)

The following figures present two timed automata (modeled in UPPAAL) that describe the states of the controller of a devaporizer (*Idle*, *Devaporizing*), and the states of the devaporizer itself (*Idle*, *Empty*, *Half*, *Full*). The automata use two logical variables (*bool devaporized*, *bool finished*) and two channels (*chan empty*, *chan devaporize*). The logical variables are initially false. Note that guards use "= =" whereas assignments use "=".

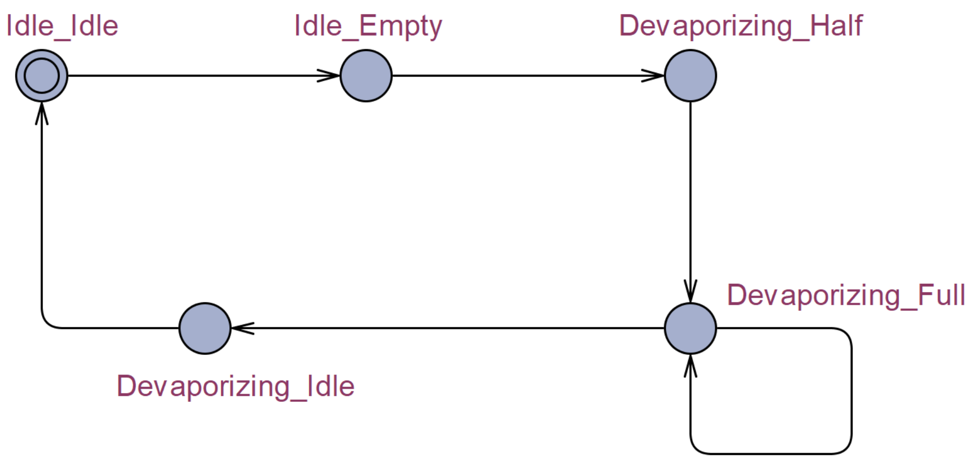
2.1. Construct the Kripke structure corresponding to the *whole system*, i.e., reachable *combinations of the states* of the controller and states of the devaporizer, and the *transitions* among the combined states. *Label* each combined state with the names of the states that it represents (you can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

5 points



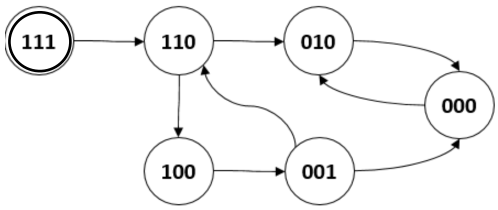


Solution:



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 → 110→ 010 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

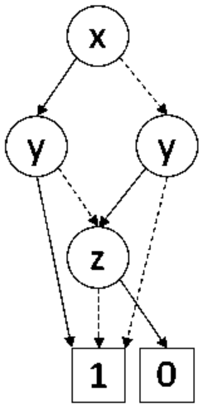
Solution:

3.1:

$$C_{111} = x \wedge y \wedge z$$
$$C_{111 \rightarrow 110 \rightarrow 010} = (x \wedge y \wedge z) \wedge (x' \wedge y' \wedge \neg z') \wedge (\neg x'' \wedge y'' \wedge \neg z'')$$

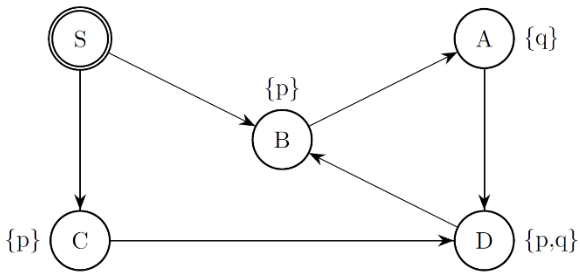
3.2:

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: **E ((AX 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:

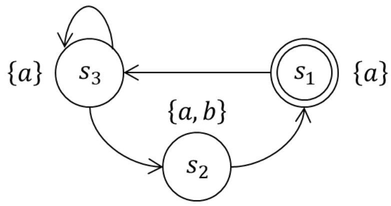
1. step: S, A, C, D states are labeled by **AX p** (all direct successor states are labeled by **p**).
2. step: A, D states are labeled by **E ((AX p) U q)** (since these states are already labeled by **q**).
3. step: C state is labeled by **E ((AX p) U q)** (this state is labeled by **AX p** and there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

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

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

In a city either *all cars* can enter, or only *electric cars*, or *none* of them (the strictness of the restriction increases in this order). The air pollution in the city can be *low* or *high* and there may be a related *alert*. We record all these facts on a daily basis.
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 city!

5.1.	If *no cars* are allowed to enter, but the air pollution is *low* and there is no *alert*, then on the next day we allow *electric* cars, and on the day afterwards we allow *all cars*.	2 points
5.2.	If there is an *alert* and *all cars* can enter or *only electric cars* can enter, then we eventually introduce a stricter rule (from *all cars* to *electric only* or *none*, or from *electric only* to *none*, respectively).	2 points
5.3.	There is an *alert* as long as air pollution is *high.*	2 points
5.4.	Use the *tableau method* to check if the requirement **¬(a U b)** 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



Megoldás:

5.1: G ((none ∧ low ∧ ¬alert) → (X electric ∧ XX all))

5.2: G (((alert ∧ all) → F(electric ∨ none)) ∧ ((alert ∧ electric) → F none))

5.3: G (alert U (¬high))

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