Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and Networks Laboratory
—— NSG, DISCO & DYNAMO ——

HS 2023 Prof. L. Josipović, Prof. L. Vanbever, Prof. R. Wattenhofer

Discrete Event Systems

Sample Solution

29th of January 2024, 15:00–17:00.

Do not open or turn before the exam starts!
Read the following instructions!

The exam takes 120 minutes and there is a total of 120 points. The maximum number of points for each subtask is indicated in brackets. **Justify all your answers** unless the task explicitly states otherwise. Mark drawings precisely.

Answers which we cannot read are not awarded any points!

At the beginning, fill in your name and student number in the corresponding fields below. You should fill in your answers in the spaces provided on the exam. If you need more space, we will provide extra paper for this. Please label each extra sheet with your name and student number.

Name	**Legi-Nr.**

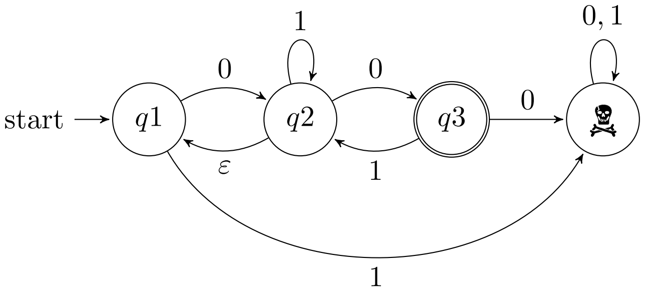
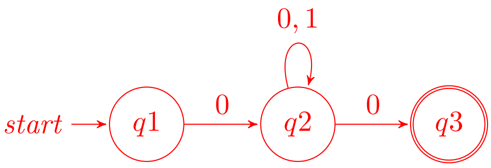
Points

#	Topic	Achieved Points	Max. Points
1	Multiple Choice on Languages & Automata		8
2	Regular Languages		12
3	Context-Free Languages		12
4	Tandem-Pumping Lemma		8
5	White Rabbit		20
6	Züri-Säcke		20
7	Equivalent BDDs		8
8	Characteristic Functions		18
9	Petri Nets		14
Total			**120**

1 Multiple Choice on Languages & Automata (8 points)

For each of the following statements, indicate whether they are **TRUE** or **FALSE**. No justification is needed. There is always one correct answer. Each block of questions is awarded up to 4 points: 4 points for 4 correct answers, 2 points for 3 correct answers, and 0 points otherwise.

1.1 Regular Languages `[4 points]`

Let $\Sigma = \{0,1\}$ and consider the automaton A:  start q1 q2 q3 ☠ 0 1 0 0, 1 0 ε 1 1	**TRUE**	**FALSE**
a) The given automaton A is a DFA. *There is an ε-transition allowing to go to states $\{q_2, q_3\}$ from q_2 upon reading a 0.*	□	☑
b) The given automaton A is irreducible. *Since this is an NFA, the ☠-state can be safely deleted. The minimal version of it looks like this:* 	□	☑
c) The language $L(A)$ interpreted as signed binary numbers describes all positive (first bit is 0) even numbers with at least two bits. *$L(A)$ describes all numbers whose bit representation starts (positive) and ends (even) with a 0.*	☑	□
d) $L(A) = \big(0 \cup 1^* \cup 01\big)^+0$. *The REX accepts 0, but the automaton does not.*	□	☑

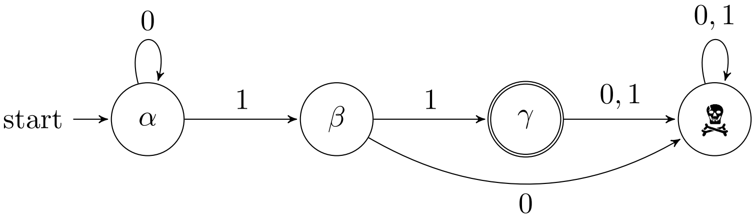
1.2 Properties of Languages [4 points]

Consider the languages on the non-empty, finite alphabet Σ: $L_{\varnothing} = \varnothing$, L_{fin}: a non-empty, finite language, L_{reg}: a regular (but infinite) language, and L_{cf}: a context-free (but irregular) language.	**TRUE**	**FALSE**
a) The language $L = L_{\varnothing} \,.\, L_{cf}$ is irregular. *$L = L_{\varnothing} \,.\, L_{cf} = \varnothing$; hence, L is finite (and thus regular).*	□	☑
b) The language $L = \overline{L_{fin}}$ must be infinite. *$\overline{L_{fin}}$ is the complement of L_{fin}, i.e., $\overline{L_{fin}} = \Sigma^* \backslash L_{fin}$. Thus, $\overline{L_{fin}}$ is regular (closure of regular languages) and infinite.*	☑	□
c) The language $L = \overline{L_{fin}} \cap L_{reg}$ is regular. *$\overline{L_{fin}}$ is regular (closure of regular languages) and infinite. In addition, regular languages are closed under intersection.*	☑	□
d) $L = \{w \mid w \in L_{fin}$ or $w \in L_{reg}$, but not both$\}$ is regular. *$L = (L_{fin} \cup L_{reg}) \cap \overline{L_{fin} \cap L_{reg}}$. Hence, L is regular due to the closure of regular languages over union, intersection and complement.*	☑	□

2 Regular Languages (12 points)

The languages L_1, L_2, L_3 are defined over the alphabet $\Sigma = \{0, 1\}$ as follows:

- $L_1 = 01^*$,
- $L_2 = (01)^*$.
- L_3 is the language recognized by the following DFA:



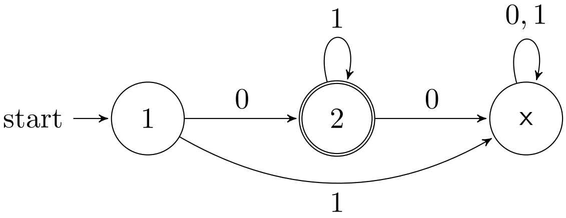
a) [2] Draw a Deterministic Finite Automaton (DFA) recognizing L_1 with at most 3 states.

b) [2] Draw a Deterministic Finite Automaton (DFA) recognizing L_2 with at most 3 states.

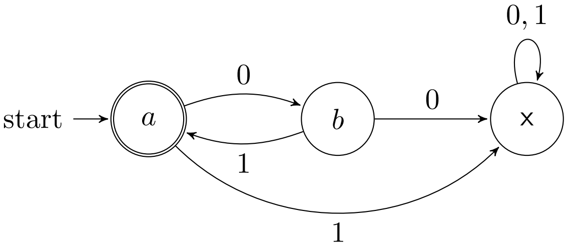
c) [8] Draw a Non-Deterministic Finite Automaton (NFA) accepting all words that are contained in **exactly one** (but not two or three) of the three languages L_1, L_2, or L_3.

Model solution

a) $L_1 = 01^*$:

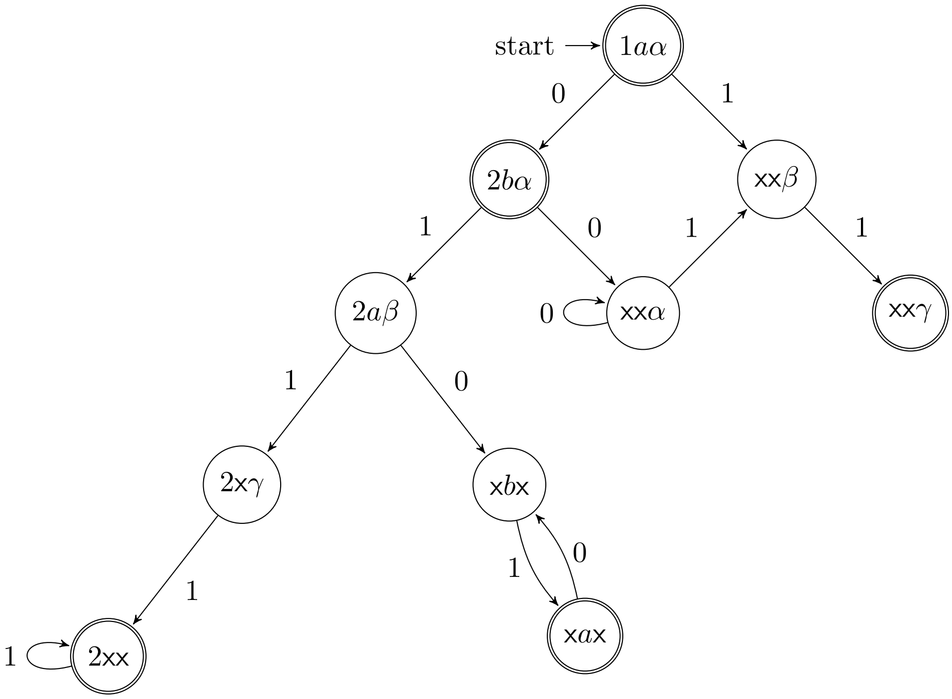


b) $L_2 = (01)^*$:



c) The cartesian product construction allows us to track the state of two automata in parallel. By applying the construction twice, i.e., to build the automaton $A = (L_1 \times L_2) \times L_3$, we obtain an automaton whose states track the three states within each automaton. We obtain the desired automaton by marking all states as accepting, that would be accepting states in exactly one of the original automata.

We build a cartesian product automaton of the three automata, using only the reachable states by iteratively building the next state:



3 Context-Free Languages (12 points)

a) [4] Give the alphabet Σ and a Context-Free Grammar (CFG) for the language L_1 using at most two non-terminal symbols:

$$L_1 = \{a^m\#b^n\#c^n\#a^m \mid m, n \geqslant 0\}.$$

b) [8] Draw a Push-Down Automaton (PDA) that recognizes L_2 with at most 8 states:

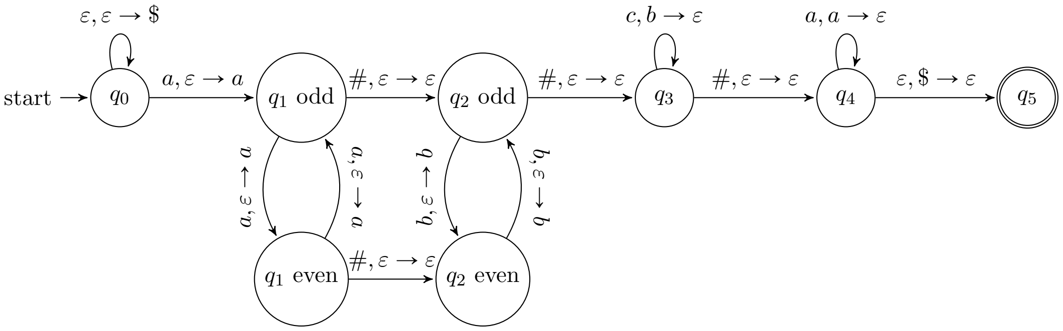
$$L_2 = \{a^m\#b^n\#c^n\#a^m \mid m > 0, n \geqslant 0, (m+n) \text{ is odd}\}.$$

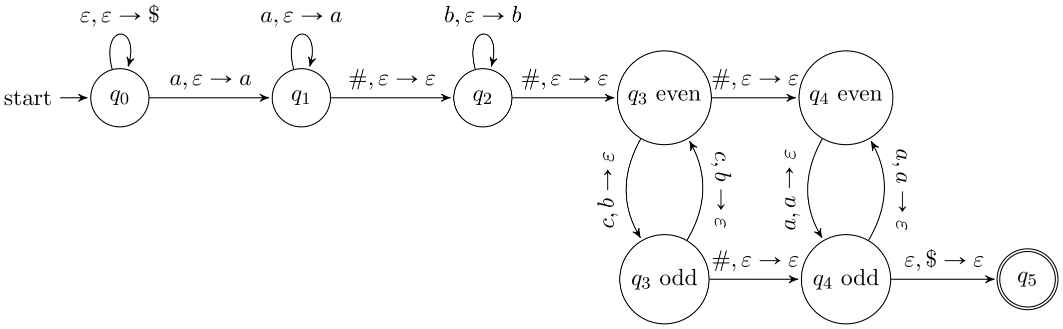
Model solution

a) The language L_1 is defined over the alphabet $\Sigma = \{a, b, c, \#\}$ and can be generated from this context-free grammar:

$$\begin{aligned} S &\to aSa \mid \#X\# \\ X &\to bXc \mid \# \end{aligned}$$

b) The language L_2 is defined over the alphabet $\Sigma = \{a, b, c, \#\}$ and can be recognized, for example, by these two push-down automata:





4 Tandem-Pumping Lemma (8 points)

Consider the language

$$L = \{a^i b^j c^k \mid 0 < i < j < k\}.$$

Show that L is not context-free using the tandem-pumping lemma.

Model solution

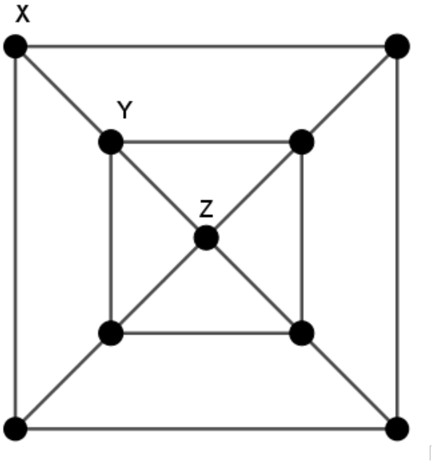
We prove that L is not context-free using the tandem-pumping lemma:

1. Assume for contradiction that L was context-free.
2. There must exist some p, s.t. any word $w \in L$ with $|w| \geqslant p$ is tandem-pumpable.
3. Choose the word $w = a^p b^{p+1} c^{p+2} \in L$ with length $|w| > p$.
4. Consider all ways to split $w = uvxyz$ s.t. $|vxy| \leqslant p$ and $|vy| \geqslant 1$.
   (i) $a \in vy$: Note that: $a \in vy \implies c \notin vy \quad$ due to $|vxy| \leqslant p$.
   We argue that $w' = uv^2xy^2z \notin L$,
      - either because $b \notin vy$ and then $\#_a(w') \geqslant \#_b(w')$,
      - or because $b \in vy$, but then $\#_b(w') \geqslant \#_c(w')$.

   (ii) $a \notin vy$: Observe that $w' = uv^0xy^0z \notin L$,
      - either because $b \in vy$ and then $\#_a(w') \geqslant \#_b(w')$,
      - or because $b \notin vy$, but then $c \in vy$ and thus $\#_b(w') \geqslant \#_c(w')$.
5. All cases combined lead to a contradiction to p being a valid tandem-pumping length.
6. Consequently, L cannot be context-free. □

5 White Rabbit (20 points)

A white rabbit is jumping around the vertices of the following garden labyrinth:



At the start, he jumps *happily* around in the following way: in every time step, independently of previous steps, he chooses a neighboring vertex of the labyrinth uniformly at random and jumps to it.

a) [2] Is the Markov Chain corresponding to the rabbit's happy jumping:

(i) irreducible YES NO

(ii) aperiodic YES NO

b) [2] What is the expected number of time steps between two successive visits to the vertex Z?

c) [4] If the rabbit starts jumping in the vertex X, what is the expected amount of time before he leaves the set of vertices $\{X, Y\}$?

d) [3] A brown rabbit enters the labyrinth. Show that if the white rabbit started in the vertex X and the brown rabbit in the vertex Y and both jump around happily (they always jump at the same time), if they eventually met in the same **vertex**, then at least one of them must have visited the vertex Z.

The rabbit gets bored and starts jumping around *smartly*: in every time step, independently of previous steps, it chooses a neighboring vertex of the labyrinth that isn't the vertex he came from in the previous time step uniformly at random and jumps to it.

e) [9] If the rabbit starts jumping smartly in the vertex Z and assumes that he came from the vertex Y, what is the expected amount of time before it leaves the inner 5 vertices of the labyrinth?

Model solution

a) The chain is irreducible, as it's a random walk on a connected graph, and aperiodic as the graph contains cycles of length 3.

b) For a random walk on a connected graph, we can compute the stationary distribution as $\pi_v = \frac{\deg(v)}{2m}$ and as the expected return time to a state Z is π_Z^{-1} we have

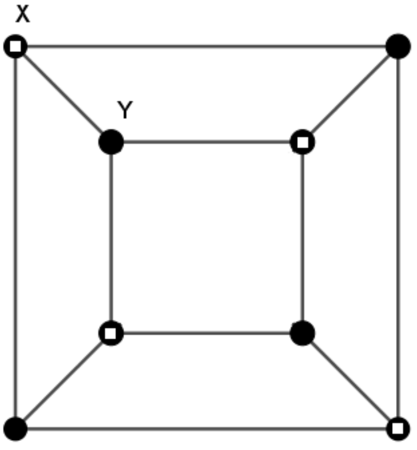
$$h_{ZZ} = \mathbb{E}[T_Z] = \frac{1}{\pi_Z} = \frac{2 \cdot 16}{4} = 8.$$

c) We denote by $\mathbb{E}_X$ the expected amount of time the rabbit takes to leave the set if starting in vertex X, and $\mathbb{E}_Y$ analogously. We then have:

$$\begin{aligned} \mathbb{E}_X &= 1 + \frac{1}{3}\mathbb{E}_Y \\ \mathbb{E}_Y &= 1 + \frac{1}{4}\mathbb{E}_X \end{aligned}$$

which gives $\mathbb{E}_X = \frac{16}{11}$.

d) Assume the opposite. Without the vertex Z, the resulting graph is bipartite (as in the picture below) and the Markov Chains corresponding to the movement of the white and brown rabbits are 2-periodic.



Now, notice that the brown rabbit starts in an unmarked vertex and always moves to a marked one while the white rabbit does the opposite, so they will never meet.

e) We first simplify the graph using its inherent symmetry. Keep the vertex Z as-is, take the other 4 interior vertices to be one new vertex I, and take the 4 exterior vertices to be one new vertex E. As the behavior of the rabbit "remembers" the last two time steps, our set of states will be ordered pairs of vertices Z, I and a state E which represents entering the outer 4 vertices. We can compute the transition probabilities as follows:

- $p_{IZ,ZI} = 1$ as after visiting Z the rabbit must go to I.
- $p_{ZI,II} = \frac{2}{3}, p_{ZI,E} = \frac{1}{3}$ as the rabbit cannot go back to Z.
- $p_{II,IZ} = \frac{1}{3}, p_{II,II} = \frac{1}{3}, p_{II,E} = \frac{1}{3}$.

We now define $\mathbb{E}_{IZ}, \mathbb{E}_{ZI}, \mathbb{E}_{II}$ analogously to task d) and we have

$$\begin{aligned}
\mathbb{E}_{IZ} &= 1 + \mathbb{E}_{ZI} \\
\mathbb{E}_{ZI} &= 1 + \frac{2}{3}\mathbb{E}_{II} \\
\mathbb{E}_{II} &= 1 + \frac{1}{3}\mathbb{E}_{IZ} + \frac{1}{3}\mathbb{E}_{II}
\end{aligned}$$

which gives $\mathbb{E}_{IZ} = \frac{9}{2}$.

6 Züri-Säcke (20 points)

You live in a shared flat and it's your turn to pay for the garbage bags ("Züri-Säcke") for the coming week. During this time, your (potentially adversarial) flatmates will generate a cumulative volume $v > 0$ of waste, but they care about the environment and you know that $v \leqslant 4$ (for simplicity assume that you do not generate any waste yourself). You don't know v in advance (except that $0 < v \leqslant 4$) and you have no information about the arrival time of the waste throughout the week.

You are in charge of buying and replacing the bags when they are full. At the start of the week, you place a new empty bag of your choice, and at the end of the week, the garbage truck will collect all perfectly and partially filled bags. Suppose that you are able to fully utilize the volume of each bag (i.e., if a piece of waste is too big to fit inside a partially filled bag, it can always be cut in a way that perfectly fills the old bag and partially fills a new one), but you will never take a piece of garbage out of a bag again ("yuck!"). There are 3 types of bags at your disposal, each in infinite quantities:

Bag type	Cost	Size
small	1	1
medium	2	2
large	3	4

a) [2] Show that there is never any gain from using a medium bag.

b) [3] What is the optimal offline strategy to select bags during the week for a given input $0 < v \leqslant 4$, and what is the cost of this strategy?

What are the competitive ratios of the two following online strategies?

c) [3] Algorithm A_s: Only use small bags.

d) [3] Algorithm A_l: Only use large bags.

e) [3] Is there another deterministic strategy that is better than these two for some $0 < v \leqslant 4$? Justify your answer.

You suspect that your flatmates do not have your best interest at heart, so you want to use a randomized strategy in order to be unpredictable.

f) [6] Derive a lower bound β on the expected competitive ratio of any *randomized* strategy. (We award full points for $\beta \geqslant 5/4$, and already 4 points for any $\beta > 1$; if you find the optimal β, we give you 3 bonus points!)

Model solution

a) You can always replace a medium bag with two small bags for the same cost and same total volume.

b) An optimal strategy is to use the smallest bag whose size is at least v. The cost of the strategy is the cost of this bag, i.e., $\min(\lceil v \rceil, 3)$.

c) The worst case happens when one could have used 1 large bag instead of 4 small bags (i.e., when $3 < v \leqslant 4$), and the corresponding competitive ratio is $\frac{4}{3}$.

d) The worst case happens when one could have used 1 small bag instead of 1 large bag (i.e., when $0 < v \leqslant 1$), and the corresponding competitive ratio is 3.

e) No. There are only two candidates for best deterministic algorithm covering a total volume of 4 (recall that you can neglect medium bags):

- A_s: The algorithm starts with a small bag. In that case, it's always profitable to continue with small bags as it will cost at most 4, while following with a large bag at any point will always cost at least 4. The competitiveness of that algorithm is 1 for $0 < v \leqslant 3$ and $\frac{4}{3}$ for $3 < v \leqslant 4$.
- A_l: The algorithm starts with a large bag and no other bag will ever be needed. The competitiveness of that algorithm is 3 for $0 < v \leqslant 1$, $\frac{3}{2}$ for $1 < v \leqslant 2$ and 1 for $2 < v \leqslant 4$.

Any other deterministic algorithm is at least as costly for every input $0 < v \leqslant 4$ (thus also in expectation).

f) Recall Yao's principle for competitive ratios: let r_r be the expected competitive ratio of any randomized online algorithm, A_{opt} the optimal offline algorithm, A_d any deterministic online algorithm and p any distribution on input v. Then

$$r_r \geqslant \min_{A_d} \mathbb{E}_{v \sim p}\left[r_d(v)\right] \quad \text{with} \quad r_d(v) = \frac{cost_{A_d}(v)}{cost_{A_{opt}}(v)}.$$

Let $p(v)$ be a distribution on $]0,4]$. By Yao's principle and using the observation of the previous question (i.e., the best deterministic algorithm is either A_s or A_l), one can simply define a distribution $p(v)$ over $]0,4]$ and propose

$$\beta = \min_{d \in \{s,l\}} \mathbb{E}_{v \sim p}\left[r_d(v)\right] = \min\left(p_{03} + \frac{4}{3}p_{34},\ 3p_{01} + \frac{3}{2}p_{12} + p_{24}\right)$$

as a lower bound, where $p_{ab} = \mathbb{P}[a < v \leqslant b]$ (e.g., the uniform distribution yields $\beta = \frac{13}{12}$). Full points are granted if $\beta \geqslant \frac{5}{4}$ (e.g., with $p_{01} = p_{34} = \frac{1}{2}$).

For the ambitious student, we now find p that yields the best possible lower bound that can be found using Yao's principle. Since we are only interested in expected performance of these two deterministic algorithms, we do not need to test all the different distributions

$p(v)$, as only the probability masses p_{01}, p_{12}, p_{23} and p_{34} are susceptible to influence the expected performance of these algorithms. Thus we have:

$$\mathbb{E}_{v\sim p}[r_s(v)] = p_{01} + p_{12} + p_{23} + \frac{4}{3}p_{34}, \qquad \mathbb{E}_{v\sim p}[r_l(v)] = 3p_{01} + \frac{3}{2}p_{12} + p_{23} + p_{34}.$$

To find the best bound, we want to maximize the minimum of these two quantities. There is clearly no benefit to put any probability mass into p_{12} and p_{23}, as this probability mass would only improve the bound if it was moved to p_{01} and p_{34}, respectively. Substituting $p_{01} = 1 - p_{34}$, we get

$$\mathbb{E}_{v\sim p}[r_s(v)] = 1 + \frac{p_{34}}{3}, \qquad \mathbb{E}_{v\sim p}[r_l(v)] = 3 - 2p_{34}.$$

Since one is increasing and the other is decreasing in p_{34}, the best bound is achieved when these two quantities are equal. Solving for p_{34} and substituting back, we get $p_{01} = \frac{1}{7}$ and $p_{34} = \frac{6}{7}$. This gives us the best lower bound $\frac{9}{7} \approx 1.28$, which is very close to the naive deterministic algorithm of always using small bags (≈ 1.33).

7 Equivalent BDDs (8 points)

Figure 1 describes the *reduced ordered binary decision diagrams* (ROBDDs) of two Boolean functions f and g; ROBDD(f) is the ROBDD of f for the variable order $\Pi_f : x \prec y \prec z$; ROBDD($g$) is the ROBDD of g for the variable order $\Pi_g : z \prec y \prec x$.

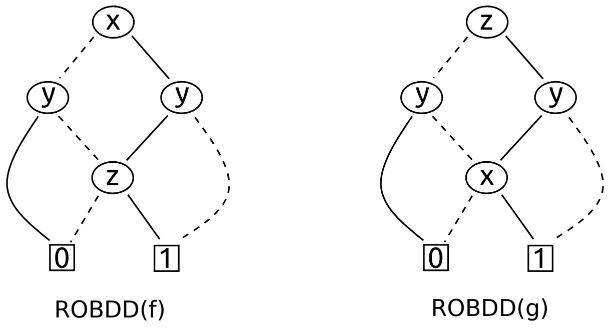


Figure 1: BDDs for Boolean functions f and g.

a) [3] How can ROBDD comparison be used to check whether two functions are equivalent? Describe in 1–2 sentences.

b) [3] Use ROBDD comparison to determine whether the Boolean functions f and g are equivalent. Clearly show the steps of your work (e.g., draw any required BDDs and BDD transformations) and justify your proof.

c) [2] Determine the logic function and draw the ROBDD of $\exists z : f(x, y, z)$ for variable order $x \prec y$.

Model solution

a) Two Boolean functions are equivalent iff the two ROBDDs, built from the two Boolean functions using **APPLY** and **SIMPLIFY** with the same variable ordering, are identical.

b) We first determine the Boolean function of g: $g(x, y, z) = x \cdot y \cdot z + x \cdot \overline{y} + \overline{x} \cdot \overline{y} \cdot z$. We use the **APPLY** operation, following the same variable order Π_f, to obtain the OBDD; we then use the **SIMPLIFY** operation to obtain the ROBDD; the obtained ROBDD is identical to the one in Figure 1a. Since the two ROBDDs are identical, therefore, the Boolean functions f and g are identical.

c) $(\exists x : f(x, y, z)) = x + \overline{y}$. See the ROBDD in Figure 2.

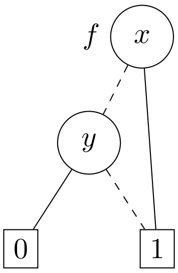


Figure 2: ROBDD of $(\exists x : f(x, y, z))$.

8 Characteristic Functions (18 points)

Figure 3 depicts a system of 3 light bulbs; there is 1 push button under each light bulb. The left side of Figure 3 describes the *initial state* of the system—light bulb 1 is in the *on* state, and light bulbs 2 and 3 are in the *off* state. A state of this system is represented using a 3-bit–binary encoding with state bits x, y, z. If light bulb 1 is in the *on* state, $x := 1$, otherwise, $x := 0$; the encodings for y and z are defined analogously. For instance, the initial state *on*, *off*, *off* is encoded as $(x, y, z) := (1, 0, 0)$.

In each state, we move to the next state by pushing *one and exactly one button*: whenever a button is pushed, the light bulb is directly above it and the adjacent light bulbs are toggled. For example, suppose that we are in the current state $(x, y, z) := (1, 0, 0)$; if we push button 3, the states of the light bulbs of the next state will be *on, on, on*, i.e., $(x', y', z') := (1, 1, 1)$.

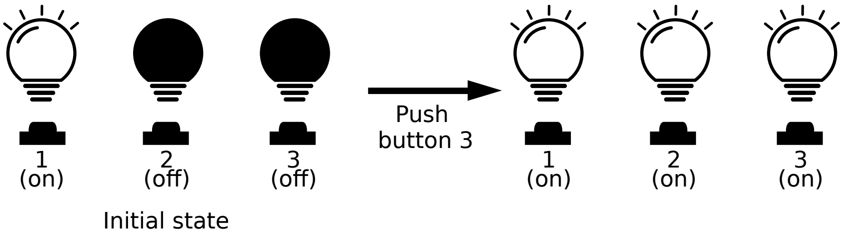


Figure 3: A system with 3 light bulbs.

Consider the state machine described above. The state space of the state machine is denoted as S and its set of state transitions as $T : S \times S$. The *characteristic function* of T is denoted as $\psi_T(x, y, z, x', y', z')$, where x, y, z are the state variables of the current state and x', y', z' are the state variables for the next state. In the following questions, when writing the characteristic functions, use only Boolean literals (e.g., x and x') and Boolean operators; when writing CTL formulas, use only Boolean literals, Boolean operations, and CTL quantifiers.

a) [3] Determine the characteristic function $\psi_1(x, y, z)$ of the set of states that satisfy the following constraint: *if light bulb 1 is in the* on *state, then exactly one of the other light bulbs, 2 or 3, must be in the* on *state.* Use only Boolean operators.

b) [3] Formulate the following property using CTL: *There is always some time in the future that the number of light bulbs that are in the* on *state is an even number (recall that zero is an even number).*

c) [3] Determine $|T|$—the number of transitions in T (both reachable and unreachable). **Justify your answer**.

d) [3] Determine the value of $\psi_T(1,0,1,0,0,0)$.

e) [3] Consider the set of initial states $Q_0 := \{(1,0,0)\}$ with the characteristic function ψ_{Q_0}. Determine the characteristic function $\psi_2(x,y,z,x',y',z')$ of the set of state transitions starting from any state in the set of initial states, i.e., $\psi_{Q_0}(x,y,z) \cdot \psi_T(x,y,z,x',y',z')$.

f) [3] Assume that you perform one step of reachability analysis, starting from the set of initial states Q_0. Determine ψ_{Q_1}—the characteristic function of the set of all reached states Q_1, i.e., $\psi_{Q_0}(x',y',z') + (\exists x,y,z : \psi_{Q_0}(x,y,z) \cdot \psi_T(x,y,z,x',y',z'))$.

Model solution

a) $\psi_1(x, y, z) = \overline{x} + (y \cdot \overline{z} + \overline{y} \cdot z)$.

b) **AF** $(\overline{x} \cdot \overline{y} \cdot \overline{z} + x \cdot \overline{y} \cdot z + x \cdot y \cdot \overline{z} + \overline{x} \cdot y \cdot z)$.

c) There are 8 states in the state space S; in each state, we push 1 of the 3 buttons, therefore, there are 24 state transitions.

d) When the current state $(x, y, z) = (1, 0, 1)$,

- when we push button 1, we get $(0, 1, 1)$
- when we push button 2, we get $(0, 1, 0)$
- when we push button 3, we get $(1, 1, 0)$

Therefore, this transition is not in the set of state transitions, i.e., $\psi_T(1, 0, 1, 0, 0, 0) = 0$.

e) The set of transitions starting with any state in Q_0 contains 3 transitions:

- $(x, y, z, x', y', z') := (1, 0, 0, 0, 1, 0)$
- $(x, y, z, x', y', z') := (1, 0, 0, 0, 1, 1)$
- $(x, y, z, x', y', z') := (1, 0, 0, 1, 1, 1)$

The characteristic function of this set evaluates to 1 iff the input is one of the 3 transitions, i.e., $\psi_2(x, y, z, x', y', z') := x \cdot \overline{y} \cdot \overline{z} \cdot (\overline{x'} \cdot y' \cdot \overline{z'} + \overline{x'} \cdot y' \cdot z' + x' \cdot y' \cdot z')$.

f) $\psi_{Q_1}(x', y', z') := (x' \cdot \overline{y'} \cdot \overline{z'} + \overline{x'} \cdot y' \cdot \overline{z'} + \overline{x'} \cdot y' \cdot z' + x' \cdot y' \cdot z')$.

9 Petri Nets (14 points)

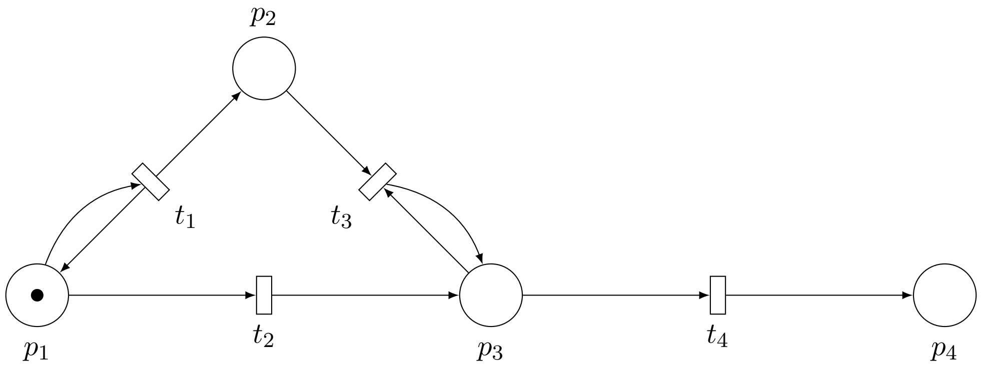


Figure 4: A Petri net.

a) [8] Consider the Petri net depicted in Figure 4. Determine the highest liveness level for transitions t_1, t_2, t_3, and t_4. **Justify your answer** for each liveness level that you determine; if the transition is L_i live but not L_{i+1}-live, explain why.

Note: A transition t in a Petri net is

- dead iff t cannot be fired in any firing sequence,
- L_1-live iff t can be fired at least once in some firing sequence,
- L_2-live iff, $\forall k \in N^+$, t can be fired at least k times in some firing sequence,
- L_3-live iff t appears infinitely often in some infinite firing sequence,
- L_4-live iff t is L_1 live for every marking that is reachable from M_0.

L_{j+1} liveness implies L_j liveness.

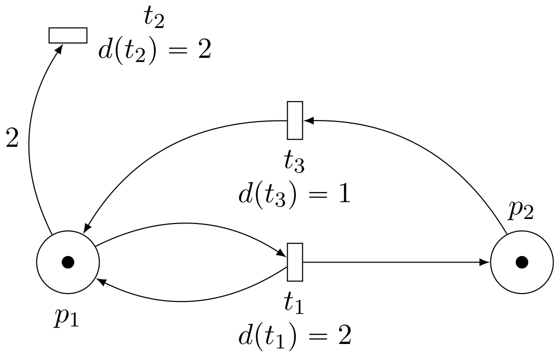


Figure 5: A time Petri net at simulation step 0 ($\tau = 0$).

b) [6] Consider the time Petri net in Figure 5 at simulation step 0. The transitions are associated with the following delays between their activation and firing: $d(t_1) = 2$, $d(t_2) = 2$, $d(t_3) = 1$. The edge (p_1, t_2) has an associated weight of 2; each other edge has a weight of 1.

Simulate the behavior of the time Petri net by filling in the table below. For each simulated step, corresponding to a firing of the Petri net, indicate the simulation time τ, the transition t_{fired} that fires in τ, the resulting marking M^τ, and the updated event list L^τ.

Note: If there are several transitions enabled at the same time, they fire in the order of their index, i.e., the transition with the smallest index fires first.

step	τ	t_{fired}	M^τ	L^τ
0	0	-	$[1, 1]$	(t_1, 2), (t_3, 1)
1				
2				

Model solution

a)

- t_1: L_3-live. We can fire t_1 infinitely starting from the initial marking, e.g., with sequence $\{t_1, t_1, \dots\}$. t_1 is not L_4-live, because t_1 is dead for any marking we obtain after firing t_2.
- t_2: L_1-live. It can be fired once and exactly once starting from the initial state, e.g., with sequence $\{t_2\}$. After firing it, there is no way to place a token at p_1, therefore it is not L_2-live.
- t_3: L_2-live. For any positive integer N, we can first fire t_1 for N times, then fire t_2 once, then fire t_3 N times. It is not L_3-live, since infinitely firing t_1 means t_2 is never fired, therefore infinitely firing t_1 means t_3 is never fired, and t_3 cannot be infinitely fired.
- t_4: L_1-live. It can be fired once and exactly once starting from the initial state, e.g., with sequence $\{t_1, t_2, t_4\}$. It is not L_2-live, since after it has been fired once, it can never be enabled again.

b)

step	τ	t_{fired}	M^τ	L^τ
0	0	-	$[1, 1]$	(t_1, 2), (t_3, 1)
1	1	t_3	$[2, 0]$	(t_1, 2), (t_2, 3)
2	2	t_1	$[2, 1]$	(t_1, 4), (t_2, 4), (t_3, 3)