



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Computer Engineering and
Networks Laboratory

— NSG, DISCO & DYNAMO —

HS 2024

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

Discrete Event Systems

Sample Solution

Monday, 10th of February 2025, 13:30–15:30.

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	Recognizing Strings with Typographical Errors		14
3	Robot Guard		18
4	Infected Graphs		20
5	TCP Acknowledgment		20
6	Binary Decision Diagrams and Model Checking		16
7	Liveness Properties and CTL of Petri Nets		16
8	Time Petri Nets		8
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]

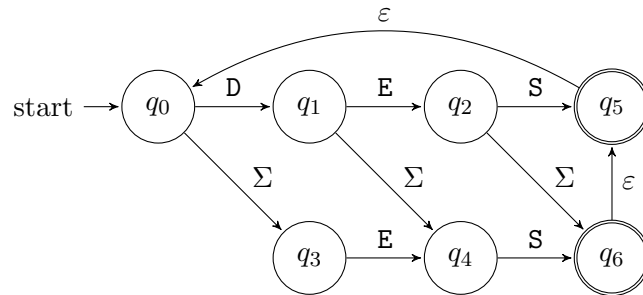
Consider the languages over the alphabet $\Sigma = \{0, 1\}$: • $L_1 = 1^*$ • $L_2 = (0 \cup 1)^*0(0 \cup 1)^*$ • L_3 an <i>arbitrary</i> irregular language defined on Σ.	TRUE	FALSE
a) The languages $\overline{L_1}$ and L_2 are equivalent. <i>$\overline{L_1}$ recognizes all words but $\{\epsilon, 1, 11, 111, \dots\}$. In other words, any word with at least one 0, which is exactly what L_2 recognizes.</i>	☒	☐
b) L_2 and $L = (0 \cup 1)^*0 \cup 0(0 \cup 1)^*$ are equivalent. <i>$101 \in L_2$, but $101 \notin L$.</i>	☐	☒
c) The language $L = \overline{L_3} \cup L_3$ is always regular. <i>$L = \Sigma^*$ recognizes all strings, which is always regular.</i>	☒	☐
d) The language $L = \overline{L_1} \cap L_3$ is always irregular. <i>Let $L_3 = \{1^p \mid p \text{ prime}\}$, then $L = \overline{L_1} \cap L_3 = L_2 \cap L_3 = \emptyset$ (because L_3 does not contain words containing a 0), which is regular.</i>	☐	☒

1.2 Context-Free Languages [4 points]

Let $\Sigma = \{a, b, +, \cdot, =, (,)\}$ and consider the grammar G: $S \rightarrow T = T$ $T \rightarrow T \cdot T \mid T + T \mid (T) \mid a \mid b$	TRUE	FALSE
a) $L(G)$ contains the binomial formula $(a + b) \cdot (a + b) = a \cdot a + a \cdot b + b \cdot a + b \cdot b$	☒	☐
b) $L(G)$ is regular. <i>$L(G) \cap \{(^na)^n \mid n \geq 0\} = \{(^na)^n \mid n \geq 0\}$, which is known to be irregular from the lecture (see $\{1^n 0 2^n \mid n \geq 0\}$).</i>	☐	☒
c) The grammar G is ambiguous. <i>The term $a \cdot a \cdot a = a$ has at least two leftmost derivations:</i> <i>$S \Rightarrow T = T \Rightarrow T \cdot T = T \Rightarrow A \cdot T = T \Rightarrow a \cdot T = T \Rightarrow a \cdot T \cdot T = T \Rightarrow \dots \Rightarrow a \cdot a \cdot a = a$, or</i> <i>$S \Rightarrow T = T \Rightarrow T \cdot T = T \Rightarrow T \cdot T \cdot T = T \Rightarrow A \cdot T \cdot T = T \Rightarrow a \cdot T \cdot T = T \Rightarrow \dots \Rightarrow a \cdot a \cdot a = a$.</i>	☒	☐
d) Any grammar G' in Chomsky Normal Form (CNF) with $L(G') = L(G)$ requires at least 6 non-terminal symbols. <i>In CNF, we require at least a dedicated starting symbol, and one symbol for each terminal symbol that cannot arbitrarily swap places with any other symbol. In this case, the only exchangeable symbols in Σ are $+$ and $\cdot$, as well as a and b.</i>	☒	☐

2 Recognizing Strings with Typographical Errors (14 points)

The following (**not minimized**) NFA, defined on the alphabet $\Sigma = \{A, B, \dots, Z\}$, recognizes the string **DES** any number of times and allows for one mis-typed letter (a *typo*). The transition label Σ denotes transitions for each character in Σ .



In this task, you can use transition labels Σ (as defined above) and, for example, $\Sigma \setminus \{X, Y\}$ to represent transitions for each character in Σ except for **X** and **Y**.

- a) [6] Transform the given NFA into an equivalent DFA using at most 9 states.

Hint: If you use the derandomization technique from the lecture, focus on the reachable states as seen in the exercise.

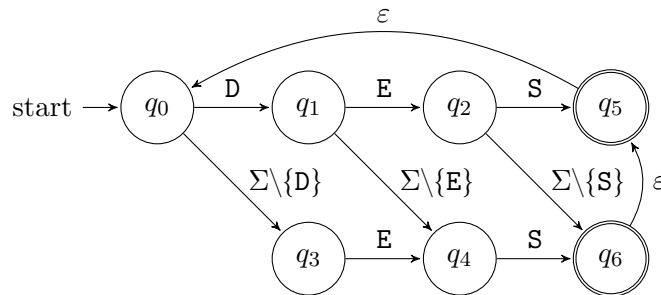
- b) [8] Draw an NFA using at most 9 states that accepts one of the words {DES, NFA, DFA} **once** with at most one *typo*.
(You will be awarded up to 4 points if your solution uses at most 12 states.)

Model solution

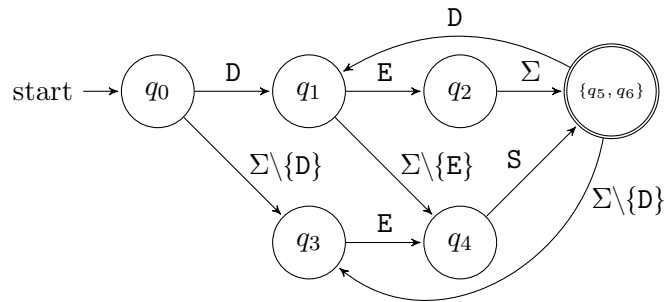
- a) We could use the systematic transformation scheme presented in the lecture. Considering the large number of states, however, this will lead to an explosion of states in the derandomized automaton ($2^7 = 128$). There are two alternative options: Either following the step-by-step derandomization procedure presented in the exercise *Derandomizing a large NFA*, or by carefully *transforming* all non-deterministic features of the given NFA into an equivalent deterministic automaton. Both alternatives yield the same solution.

We build the DFA by transforming non-deterministic features one after the other:

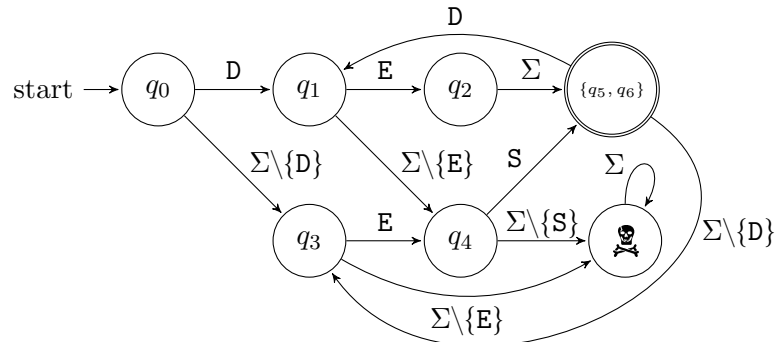
1. Starting from q_0 , we notice that the outgoing transition for D is not deterministic. We can eliminate this ambiguity by restricting the transition from q_0 to q_3 to $\Sigma \setminus \{D\}$. Similarly, the same can be done for q_1 and q_2 , yielding:



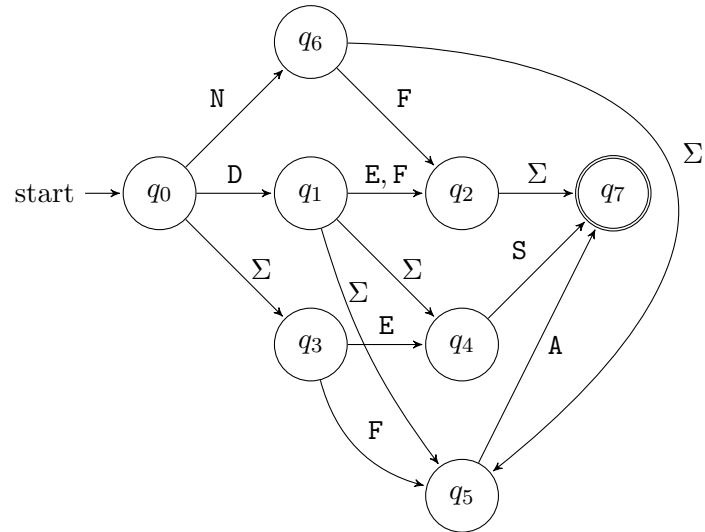
2. Next, we want to get rid of ε -transitions. Since the only outgoing transition from q_6 is the ε -transition to q_5 and both q_5 and q_6 are accepting states, we can simply merge them together. Furthermore, the transition from q_5 to q_0 can be made deterministic by reading either D and going to q_1 or reading anything else and going to q_3 .



3. Finally, we add a crash state for all undefined transitions.



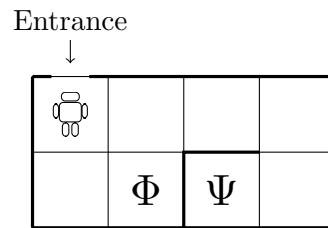
- b) Taking inspiration from the NFA drawn in the task description, we can first minimize it by removing the ε -transitions (as we now want to recognize the word once) and merging the accepting states. Then we can simply duplicate states q_1 through q_4 to recognize NFA and DFA. This would yield an NFA with 14 states. Then we can start to merge states together since the words share characters. The minimized NFA with 8 states looks like:



3 Robot Guard

(18 points)

The *Museum of Ancient Greek Letters* installs a robot guard to protect its most valuable exhibits, Φ and Ψ . Consider the museum's floor plan:



The robot can move up (u), down (d), left (l), and right (r), i.e., paths are defined on the alphabet $\Sigma = \{\mathbf{u}, \mathbf{d}, \mathbf{l}, \mathbf{r}\}$. Your task is to ensure that the robot only takes valid paths through the museum.

The language of valid paths L_{path} is defined as all paths that:

- start and end at the museum's entrance, but should never leave;
- do not hit any of the museum's walls (bold lines: $\boldsymbol{\lrcorner}$); and
- visit each of the squares of the exhibits Φ and Ψ an equal number of times.

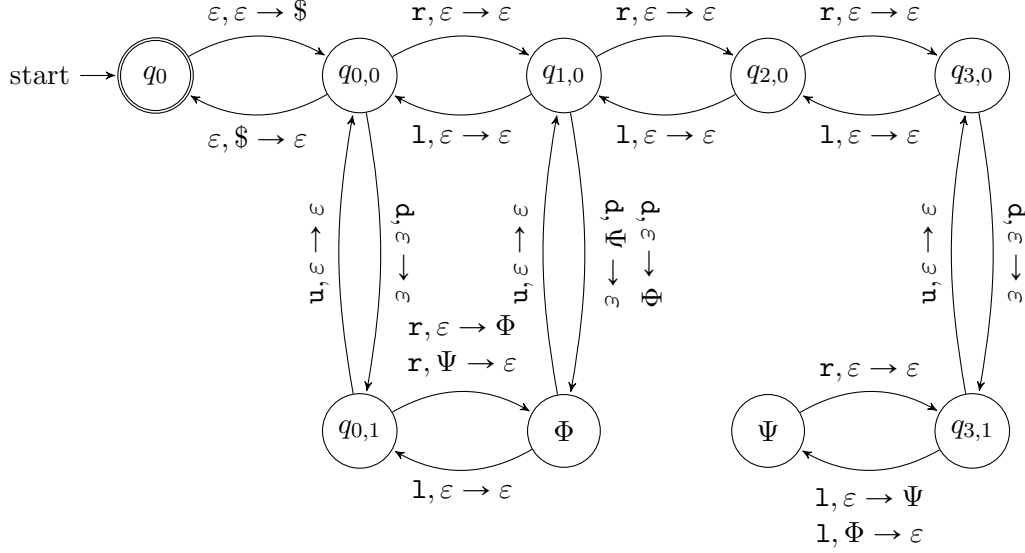
a) [8] Draw a Push-Down Automaton (PDA) that recognizes all valid paths in L_{path} .

b) [10] Prove that the language L_{path} is not regular using the pumping lemma.

Hint: Consider what happens to a pumped-up path xy^iz for $i \geq 4$.

Model solution

a) A PDA recognizing L_{path} could look like this:



b) We prove that L_{path} is not regular using the pumping lemma.

1. Assume for contradiction that L_{path} was regular.
2. There must exist some p , s.t. any word $w \in L$ with $|w| \geq p$ is pumpable.
3. Consider the word $w = r \underbrace{(\text{du})^p}_{\text{visit } \Phi} \text{rrd} \underbrace{(\text{lr})^p}_{\text{visit } \Psi} \text{ulll}$ with length $|w| > p$. Observe $w \in L$ as:

- it starts and ends at the museum entrance without ever leaving,
- it does not hit any of the museum's walls, and
- it visits each of the exhibits Φ and Ψ exactly p times.

4. Consider all ways to split $w = xyz$ s.t. $|xy| \leq p$ and $|y| \geq 1$.

As indicated in the hint, we begin by focusing on pumped-up paths xy^iz with $i \geq 4$. Note that when the pumped-up part y contains more symbols of one direction than of its reverse direction, this would eventually cause the robot to exceed the boundaries of the museum, latest after repeating this pumped-up part 4 times.

- If $\#_r(y) \neq \#_l(y)$ or $\#_d(y) \neq \#_u(y)$, then $w' = xy^4z$ cannot be in L_{path} as the robot would exceed the museum's boundaries at some point.
- Otherwise, we have $\#_r(y) = \#_l(y)$ and $\#_d(y) = \#_u(y)$.

Note that xy must be taken from w 's prefix $r(\text{du})^p$ due to $|xy| \leq p$, and thus both x and y do not contain any 1. Hence, $\#_r(y) = \#_l(y) = 0$ and $\#_d(y) = \#_u(y) > 0$. Then $w' = xy^0z$ cannot be in L_{path} as the robot would visit exhibit Φ less often than exhibit Ψ .

5. Observe that in all cases $w' = xy^iz \notin L_{path}$ for (at least) some $i \geq 0$, which is a contradiction to p being a valid pumping length.
6. Consequently, L_{path} cannot be regular. □

4 Infected Graphs

(20 points)

In this task, we will disinfect graphs. More precisely, we have an undirected graph G without parallel edges or self-loops. At the start, each vertex of G is either *clean* or *infected*. We cannot, at any point, tell whether a vertex is clean or infected. We try to clean the graph G in the following way: once every second, we pick a vertex and *spray* it, which does the following:

- If the vertex was infected, it becomes clean.
- If the vertex was clean, but has any infected neighbors, it becomes infected.
- If the vertex and all of its neighbors were clean, nothing happens.

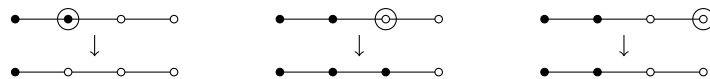


Figure 1: This picture shows each of the three cases on the path graph P_4 . Infected vertices are marked in black, clean vertices are marked in white, and the sprayed vertex is circled.

- a) [6] Even in very simple graphs, it is not always obvious how to disinfect. Let $G = P_2$, the path on two vertices. Find an order in which to spray the vertices of G (multiple times, if necessary) that guarantees the graph is clean after it is performed, no matter which vertices were infected at the start.

It can be shown that such a sequence exists for any graph, though it can be quite elaborate and tedious to construct. The main focus of this task is to show that spraying uniformly at random can in fact work rather well. Our cleaning procedure will be the following: once every second, we will pick a single vertex of the graph uniformly at random and spray it.

- b) [2] Fix some graph G . Consider a Markov chain where each state is one possible partition of the graph into clean and infected vertices, and transition probabilities are assigned by the probability of a state to change into another in the above-described process.

Is this Markov chain irreducible?

- c) [7] Consider the graph $G = C_3$, a cycle of length 3. If all vertices in the graph start out infected, what is the expected amount of time before the graph is clean?

In fact, we can show that every graph is eventually disinfected with probability 1 using the random procedure. We first show that chains of a certain type always converge to a state.

- d) [3] Consider a Markov chain with states $0, 1, 2, \dots, k$ such that $p_{0,0} = 1$ and $p_{i,0} > 0$ for all $1 \leq i \leq k$. Show that the probability of the chain eventually entering state 0 is 1, regardless of which vertex we start from.

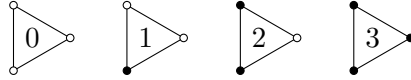
Hint: Find a p such that the chain enters state 0 with probability at least p once every unit of time. Use it!

The Markov chain of task c) is not necessarily of this form, but it can be made so with minor modifications; namely, we take n steps at once instead of just one.

- e) [2] Fix an n -vertex graph G . Show that regardless of which vertices are infected at the start, there is a non-zero probability that G is disinfected after exactly n sprays.

Model solution

- a) We spray the vertices in the order 121. If the second vertex was clean, we are immediately done. If both were infected, we are done after 12, and otherwise after 121.
- b) The state where all vertices are clean is a state we cannot leave and reachable from any other, so the chain cannot be irreducible.
- c) C_3 is small and symmetric so having 1 state per possible number of infected nodes suffices.



The transition probabilities are $p_{00} = p_{32} = 1, p_{10} = p_{23} = \frac{1}{3}, p_{12} = p_{21} = \frac{2}{3}$. Let h_k be the expected time to reach 0 from k . We then have:

$$\begin{aligned} h_1 &= 1 + \frac{2}{3}h_2 \\ h_2 &= 1 + \frac{2}{3}h_1 + \frac{1}{3}h_3 \\ h_3 &= 1 + h_2 \end{aligned}$$

which gives $h_3 = 10$.

- d) Fix $p = \min_{i \in [n]} p_{i,0}$. At every step, the probability of going to 0 is at least p , so after m steps, the probability of the chain not having yet gone to 0 is at most $(1 - p)^m$. Taking $m \rightarrow \infty$, the probability of the chain never reaching 0 is 0.
- e) Spraying each infected vertex in some order disinfects the graph, and this randomly occurs with nonzero probability as there are at most n of them.

5 TCP Acknowledgment

(20 points)

Consider the TCP acknowledgment setting seen in the lecture with the following twist: packets $1, 2, \dots, n$ arrive **consistently** at fixed intervals of δ units of time (see Fig. 2 below). While you know that packets follow this pattern, the exact values of δ and n are unknown in advance. You are interested in designing an online TCP acknowledgment algorithm, given that $0 < \delta \leq 1$ and $1 \leq n < \infty$. Each acknowledgment (ACK) costs 1 and acknowledges all packets received since the previous ACK. If a packet arrives at the exact moment an ACK is sent, it is considered acknowledged. However, if a packet waits d units of time before being acknowledged, this incurs an additional latency cost of d . Assume that the first packet arrives at $t = 0$.

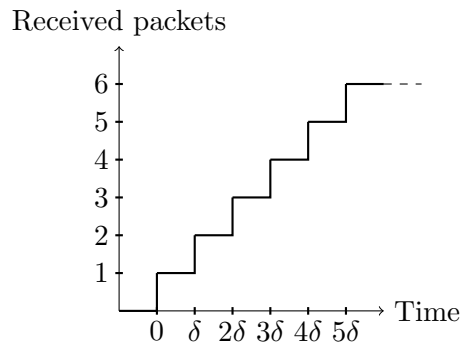


Figure 2: Packets arriving at intervals of δ with $n = 6$.

- a) [3] You come up with a simple algorithm $\mathcal{A}_p$ that sends an ACK every p packets (e.g., $\mathcal{A}_{p=2}$ sends ACKs instantly when receiving packets 2, 4, 6, etc., and at no other times). Is $\mathcal{A}_{p=4}$ a correct r -competitive algorithm for some $r < \infty$? Justify your answer.
- b) [5] Show that any optimal offline algorithm sends all its ACKs at times that are integer multiples of δ (i.e., the i th ACK is sent at time $t_i = m_i \delta$ for some integer values of $m_i \geq 0$, where $i \in \{1, 2, \dots, k\}$ are the indices of the ACKs).

- c) [5] What is the cost of an optimal offline algorithm for $\delta = 1$? Justify your answer.

From now on, assume **it is known** that $\delta = 1$ and that $n = 2^k$ for some integer $k \geq 2$ (i.e., k is the only unknown variable when you design your online algorithm).

- d) [7] Recall from the lecture the online TCP acknowledgment algorithm $\mathcal{A}_z$ which sends an ACK as soon as a rectangle of size z can fit in the latency area. What is the competitive ratio of $\mathcal{A}_{z=4}$ in this setting? Justify your answer.

Model solution

- a) No. For instance, if $n = 1$, no ACK will be sent and the latency will be unbounded.
- b) By contradiction, assume there exists an optimal offline algorithm $\mathcal{A}_\epsilon$ that sends ACK i at time $(m_i + \epsilon)\delta$ when executed on input (n, δ) , with $0 < \epsilon < 1$ and $m_i \in \mathbb{N}$. If ACK i acknowledges 0 packets, removing it simply decreases the cost by 1. Otherwise, since no packet arrive between $m_i\delta$ and $(m_i + \epsilon)\delta$, sending ACK i at $m_i\delta$ instead of $(m_i + \epsilon)\delta$ acknowledges the same packet(s) while reducing the cost by at least $\epsilon > 0$ in latency. Both cases contradict the optimality of $\mathcal{A}_\epsilon$.
- c) Since $\delta = 1$, we know from **b)** that the optimal offline algorithm only sends ACK i at integer times $m_i \in \mathbb{N}$. If a packet is not acknowledged instantly, then it will cost a latency of at least $\delta = 1$ (until the next ACK). Therefore it is never worse to acknowledge each packet instantly, which would cost n in total.
- d) The competitive ratio is given by $r_{z=4} = \sup_{2 \leq k < \infty} \frac{\text{cost}_{\mathcal{A}_{z=4}}(k)}{\text{cost}_{\text{opt}}(k)}$. Since $n = 2^k$, we know from **c)** that $\text{cost}_{\text{opt}}(k) = 2^k$. To compute $\text{cost}_{\mathcal{A}_{z=4}}(k)$, we note that $n = 2^k = 4 \cdot 2^{k-2}$ is divisible by 4. From Fig. 3, we see that each 4 consecutive packets incur a cost of 7 (6 in latency, 1 for the ACK). Hence, we have $\text{cost}_{\mathcal{A}_{z=4}}(k) = 7 \cdot 2^{k-2}$ and we can finally write:

$$r_{z=4} = \sup_{2 \leq k < \infty} \frac{7 \cdot 2^{k-2}}{2^k} = \frac{7}{4}.$$

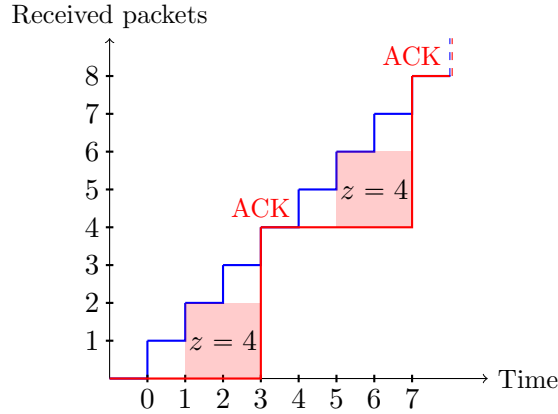


Figure 3: Visualization of the $\mathcal{A}_{z=4}$ algorithm for known $\delta = 1$.

6 Binary Decision Diagrams and Model Checking (16 points)

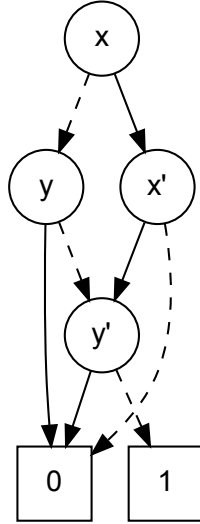


Figure 4: A BDD of $f_T(x, y, x', y')$.

Consider the finite state machine with the set of states S and the set of transition relations $T : S \times S$. The states are encoded using 2 state bits, x and y . Figure 4 shows a BDD representing the characteristic function $f_T(x, y, x', y')$ of the set of transition relations (x and y denote the current state, x' and y' denote the next state).

An arrow with a dashed line indicates that the parent node evaluates to zero; an arrow with a solid line indicates that the parent node evaluates to one.

- a) [4] Determine the characteristic function $f_T(x, y, x', y')$ from Figure 4.

- b) [4] Determine $g := (\forall y : f_T)$ and draw the **ROBDD** of g with respect to a variable order: $\Pi : y < y' < x < x'$ (consider y first, then y' , and so on).

- c) [4] Complete the state transition graph below according to the BDD in Figure 4 (i.e., draw the transition edges between the state nodes).

Note: the label on the state node indicates the assignment to the state variables (e.g., “ $x \neg y$ ” indicates that $x = 1$ and $y = 0$).



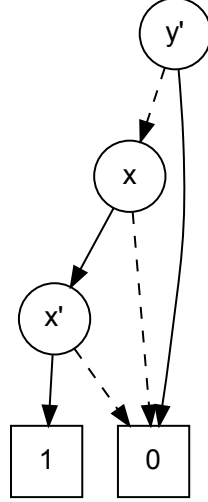
- d) [4] Consider the characteristic function $f_{Q_0}(x, y) := \neg y$ of a set of states Q_0 . Determine Q_1 , the set of all states reachable in 1 transition step from Q_0 . Have we determined the set of all reachable states? Justify your answer.

Note: Q_1 is the union between Q_0 and $Suc(Q_0)$ (the set of states that have a predecessor in Q_0).

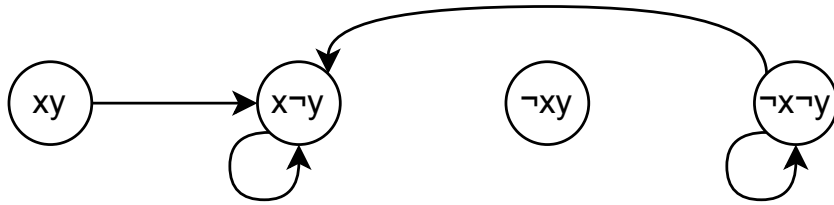
Model solution

a) $f_T(x, y, x', y') := x \cdot x' \cdot \neg y' + \neg x \cdot \neg y \cdot \neg y'$

b) $\forall y : f_T := f_T|_{y=0} \cdot f_T|_{y=1} := x \cdot x' \cdot \neg y'$. The BDD is depicted in the figure below.



c) From a) we can get all the four minterms of $f_T := x \cdot y \cdot x' \cdot \neg y' + x \cdot \neg y \cdot x' \cdot \neg y' + \neg x \cdot \neg y \cdot \neg x' \cdot \neg y' + \neg x \cdot \neg y \cdot x' \cdot \neg y'$. They correspond to four transition edges.



d) $Q_0 := \{x\neg y, \neg x\neg y\}$. $Suc(Q_0) := \{x\neg y, \neg x\neg y\}$. The set of all states reachable from Q_0 is 1-step, $Q_1 := Q_0 \cup Suc(Q_0) = \{x\neg y, \neg x\neg y\}$. Since $Q_1 \equiv Q_0$, the fixed-point of the state transformation is reached, therefore we have determined the set of all reachable states.

7 Liveness Properties and CTL of Petri Nets

(16 points)

Recall that 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, $\forall k \in \mathbb{N}^+$, 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 below depicts Petri net N_1 with 3 places (p_1 , p_2 , and p_3) and 4 transitions ($t_1, \dots, t_4$).

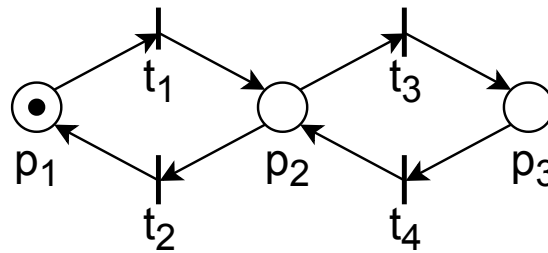


Figure 5: The Petri net N_1 for Exercise 7.

Note: Let us assume that unless the Petri net is dead (i.e., no transition is activated), *we must fire one transition*. Answer the following questions:

- [3] Consider N_1 in Figure 5. What is the highest liveness level of t_4 ? If t_4 is L_k -live but not L_{k+1} -live, demonstrate a proof (no proof is needed if you answered t_4 is L_4 -live).
- [5] Consider N_1 in Figure 5. For each transition $t_k \in \{t_1, t_2, t_3, t_4\}$, determine whether N_1 satisfies **AG AF** “ t_k is activated in the current marking”. Justify your solutions for the unsatisfied CTL formulas by providing a counterexample trace (**note:** you only need to provide “yes/no” for the satisfied formulas).

Figure 6 shows the Petri net N_2 with 3 places (p_1 , p_2 , and p_3) and 3 transitions (t_1 , t_2 , and t_3).

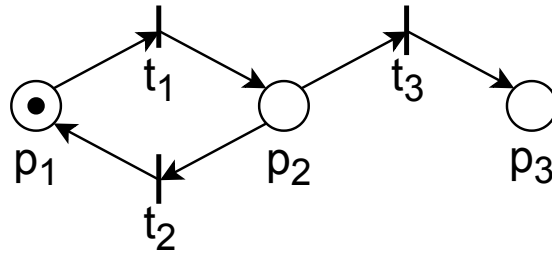


Figure 6: The Petri net N_2 for Exercise 7.

Answer the following questions:

- c) [3] Consider N_2 in Figure 6. What is the highest liveness level of t_3 ? If t_3 is L_k -live but not L_{k+1} -live, demonstrate a proof (no proof is needed if you answered that t_3 is L_4 -live).

- d) [5] Consider N_2 in Figure 6. For each transition $t_k \in \{t_1, t_2, t_3\}$, determine whether N_2 satisfies **EG EF** “ t_k is activated in the current marking”. Justify your solutions for the satisfied CTL formulas by providing a trace (**note**: here you need to justify the satisfied formulas; you only need to provide “yes/no” for the unsatisfied formulas).

Model solution

- a) t_4 is L_4 -live.
- b) To show that N_1 does not satisfy **AG AF** “ t_4 is activated”, we need to show that N_1 satisfies $\neg$ **AG AF** “ t_4 is activated”, or equivalently, **EF EG** $\neg$ “ t_4 is activated”.

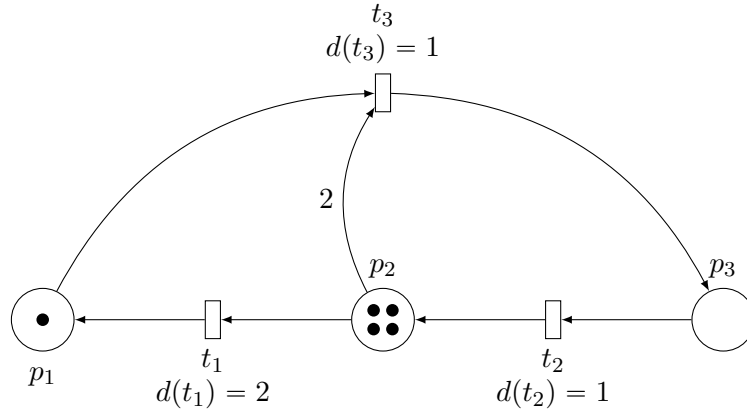
Starting the initial marking, there exists a trace that infinitely alternates firing t_1 and t_2 , such that t_4 is never activated. Thus **AG AF** “ t_4 is activated” is not satisfied.

Similarly, starting from the initial marking and after firing t_1 , there exists a trace that infinitely alternates t_3 and t_4 ; t_1 is only activated in the initial state, thus **AG AF** “ t_1 is activated” is not satisfied.

For both extreme scenarios above, the place p_2 is marked with a token infinitely often, which activates t_2 and t_3 infinitely often. Thus, N_1 satisfies **AG AF** “ t_2 is activated” and **AG AF** “ t_3 is activated”.

- c) t_3 is L_1 -live: After firing t_3 there is no way to fire it again. Therefore, it is not L_2 -live.
- d) We can prove that N_2 satisfies the CTL formulas **EG EF** “ t_1 is activated” and **EG EF** “ t_2 is activated” by providing an infinite trace: $(1, 0, 0), (0, 1, 0), (1, 0, 0), \dots$ obtained by firing t_1 and t_2 in an interleaved manner. t_1 and t_2 are activated infinitely often on this trace, thus N_2 satisfies **EG EF** “ t_1 is activated” and **EG EF** “ t_2 is activated”.

The set of reachable markings that satisfy **EF** “ t_3 is activated” are $(1, 0, 0)$ and $(0, 1, 0)$. Every state on the infinite trace above is one of them, thus, N_2 satisfies **EG EF** “ t_3 is activated”.

Figure 7: A time Petri net at simulation step 0 ($\tau = 0$).

- a) [8] Consider the time Petri net N_3 in Figure 7 at simulation step 0. The transitions are associated with the following delays between their activation and firing: $d(t_1) = 2$, $d(t_2) = 1$, $d(t_3) = 1$. The edge from p_2 to t_3 has a weight of 2; all other edges have 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: When several transitions can be fired at the same time, they fire in the order of their indices, i.e., the transition with the smallest index fires first.

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

Model solution

- a) At step = 1, firing t_3 removes a token from p_2 and therefore t_1 loses activation and reactivated immediately. At step = 2, firing t_2 adds a token to p_2 —this does not disable t_1 .

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

