Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and Networks Laboratory
TEC, NSG, DISCO

HS 2019 Prof. L. Vanbever, Prof. R. Wattenhofer, Prof. L. Thiele

Discrete Event Systems

Exam

Tuesday, 28th January 2020, 09:00–11:00.

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

Question	Topic	Achieved Points	Maximal Points
1	Regular Languages		23
2	Context-free Languages		17
3	Company Helpline		24
4	Task Allocation		16
5	True or False		6
6	Binary Decision Diagram		10
7	Petri Nets		24
Total			**120**

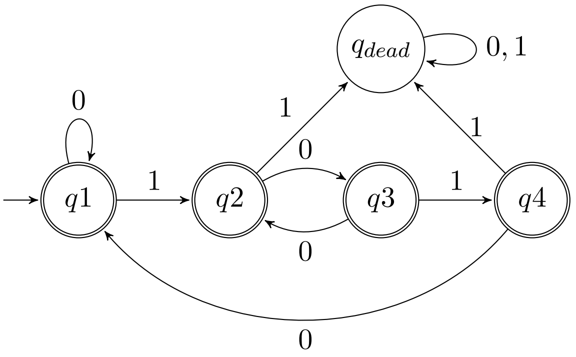
1 Regular Languages (23 points)

1.1 True or False [4 points]

For each of the following statements, indicate whether they are **TRUE** or **FALSE** (circle **one** answer). For each question answered correctly, one point is added. For each question answered incorrectly, one point is removed. No answer gives zero point. There is always one correct answer. This subtask gives at least 0 points.

a) **TRUE** **FALSE** | Consider the language {banana}. It is possible to build a DFA with an arbitrary number of states recognizing it.

b) **TRUE** **FALSE** | The following DFA recognizes a string only if there is an odd number of 0's between every subsequent pair of 1. For instance 00 and 1000101 are recognized whereas 1001 is not.



c) **TRUE** **FALSE** | The regular expression $(01^*0 \cup 1)^*0$ recognizes the language consisting of all strings over $\{0, 1\}$ having an odd number of 0's.

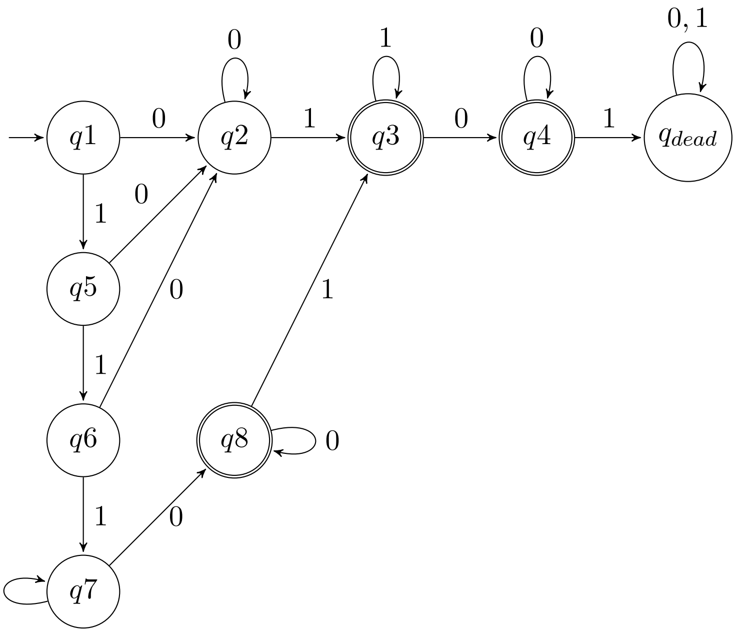
d) **TRUE** **FALSE** | In a finite language no string is pumpable.

Model solution

a) True (we accepted also False no answer here since it is not possible to find a DFA of size 1, for instance).
b) False. 010001001 is recognized.
c) False. 0001 has an odd number of 0 but is not in the language.
d) True. By definition of the pumping lemma.

1.2 The minimum pumping length `[9 points]`

Consider the following DFA D.



a) `[4]` Find a regular expression simulated by D which contains at most **one** union operator. For time efficiency, you do **not** have to compute the GNFA.

b) [5] Give the minimum pumping length of the regular language recognized by D. Briefly explain the intuition behind your answer.

Model solution

a) By observing the DFA, we can see that it recognizes the following language:
$L = 0^{+}1^{+}0^{*} \cup 10^{+}1^{+}0^{*} \cup 110^{+}1^{+}0^{*} \cup 111^{+}0^{+}1^{+}0^{*} \cup 111^{+}0^{+}$
Which can be simplified to:
$L = 1^{*}0^{+}1^{+}0^{*} \cup 111^{+}0^{+}$
Points distribution: -1 for each mistake. 0 if there is only one part, 0 if more than one union.

b) We know that the minimum pumping length of a language $L = L_1 \cup L_2$ is $p = max\{p_1, p_2\}$ where p_1 and p_2 are the minimum pumping length of L_1 and L_2, respectively.
Here, let's assume $L1 = 1^{*}0^{+}1^{+}0^{*}$ and $L2 = 111^{+}0^{+}$.
The minimum pumping length of $L2$ cannot be 4 because 1110 cannot be pumped. Now consider the string s that belongs to $L2$ and that has a size of 5. If $s = 11110$, then it can be divided into xyz where $x = 111$, $y = 1$ and $z = 0$ and thus can be pumped. If $s = 11100$, then it can be divided into xyz where $x = 111$, $y = 0$ and $z = 0$ and thus can be pumped. The minimum pumping length for $L2$ is thus 5.
A string s of size 3 and belonging to L1 can always be pumped.
To conclude, the minimum pumping length of L is 5.

Points distribution: 2 points if the reasoning makes sense but answer if false. 3 points if the reasoning is correct and the answer is correct according to the answer in the previous question. 5 points if the the answer is fully correct.

1.3 On the size of the minimized DFA `[10 points]`

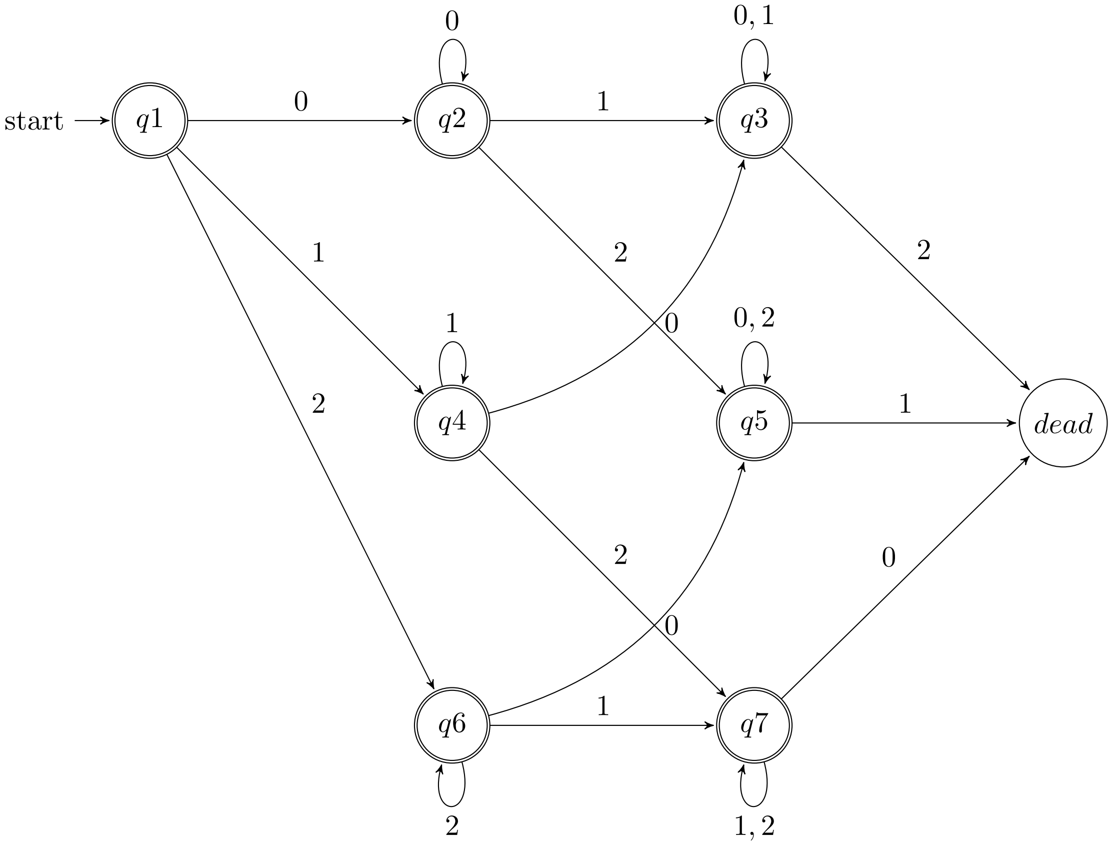
Find a language L that accepts strings over Σ where $|\Sigma| = n$ and for which its corresponding minimized DFA has a size equal to 2^n. Then, draw the minimized DFA with 8 states recognizing L when $|\Sigma| = 3$ (e.g., $\Sigma = \{0, 1, 2\}$).

Briefly explain the intuition behind your answer. You may get points even if your answer is incomplete, but an answer without explanation will give 0 points.

Hint: Start by considering a simple alphabet such as $\Sigma = \{0, 1\}$.

Model solution

The language L accepts the strings which do not contain all the symbols of Σ.



2 Context-free languages (17 points)

2.1 Is this a CFL? [7 points]

a) [4] Prove that $L_1 = \{10^n10^n10^{m+n} \mid n, m \geqslant 0\}$ is not context free.

b) [3] Prove that $L_2 = \{xxxw \mid x \in (0 \cup 1)^+, w \in (0 \cup 1)^*\}$ is not context free using the fact that context-free languages are closed under intersection with a regular language.

Model solution

a) Choose $z = 10^p10^p10^p$, $z = uvwxy$, $|vx| > 0$

- If v or x contains a 1, take $i = 0$ and uv^iwx^iy has less than three 1's and cannot be in L. In the remaining case v and x contain 0's from at most two groups.
- If $|v| = |x|$ and they take 0's from the first two groups pump up $uv^2wx^2y = 10^{n+|v|}10^{n+|v|}10^n \notin L$
- In all other cases pump down: $uwy \notin L$ either the first two groups of 0's have a different length or the last group contains less 0's than the other groups

Points

(i) (4 Points) if all three cases are explained.

(ii) (3 Points) if two cases are explained.

(iii) (0 Points) if pumping lema for regular languages is used.

b) Intersect with regular language L = 10*10*10*. It suffices to prove that the resulting language $L = \{10^n10^n10^{m+n}|n, m > 0\}$ is not context free since CFL's are closed under intersection with regular languages. We have done so in the previous question.

2.2 Draw me a PDA `[10 points]`

Construct a non-deterministic PDA with at most 15 states that recognizes the following language B defined over $\Sigma = \{0, 1, \#\}$:

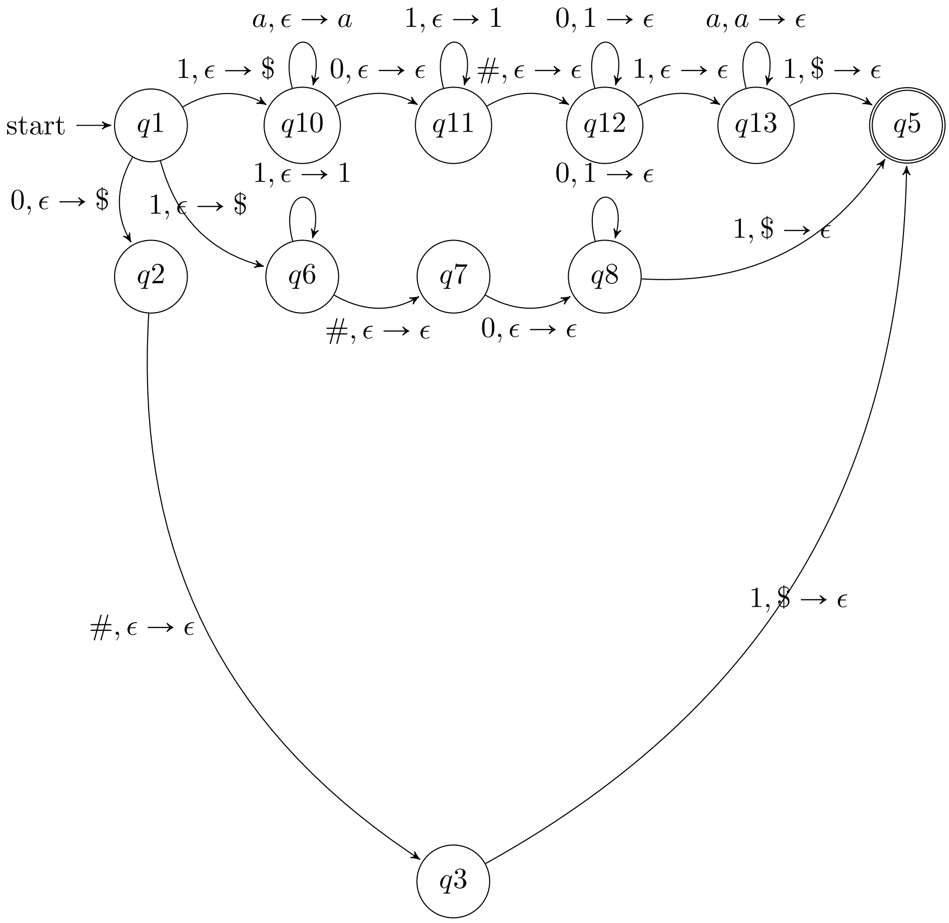
$$B = \{\mathbf{binary}(i)\#\mathbf{reverse}(\mathbf{binary}(i+1)) \mid i \in \mathbb{N}_0\}$$

where:

- **binary(x)** $\in \{0, 1\}^*$ returns the binary representation without leading zeros of the natural number x or zero if $x = 0$, e.g. **binary**$(5) = 101$.
- **reverse(y)** $\in \{0, 1\}^*$ returns the reverse representation of the binary number y, preserving leading zeros (if any), e.g. **reverse**$(1100) = 0011$.

For example, your PDA should accept the following strings: 0#1, 111#0001, 10010#11001

Model solution



Points

a) (-5 Points) if it accepts #1.

b) (-3 Points) for each of the following strings that the PDA does not accept (0#1, 1#01 , 10#11, 11#001, 101#011)

c) (full points) for PDAs with less states that were recognizing all five strings and did not accept #1, even if they few errors.

3 Company Helpline (24 points)

Your company has a helpline with 2 workers. When a customer calls the helpline, a central dispatcher immediately forwards the call to one of the call center workers, with probability p to worker 1 and probability $(1-p)$ to worker 2. The customer then waits in the respective queue until being served. Worker 1 processes requests at a rate of 1 per hour and worker 2 processes requests at a rate of 2 per hour (exponentially distributed). Incoming calls arrive according to a Poisson process with rate 1 per hour.

a) [3] At equilibrium, what is the mean time a customer spends on the phone (including both waiting and serving time), if the customer is forwarded to worker 1?

b) [9] What value of p should you choose in order to minimize the expected time a customer spends on the phone.

Unfortunately you had to replace your 2 call center workers. The new workers are less experienced. They still process requests at the same respective rates, but now they either process a request successfully with probability 1/2 or they process the request unsuccessfully and then forward it to the other worker. (A customer may be forwarded back and forth multiple times between the workers, accumulating waiting and processing times at each step. Each time the processing time is independent of previous processing times.)

c) `[7]` For what values of p is the system stable?

d) `[5]` We want to minimize the expected time a customer is in the system. Explain briefly (words or simple math) why sending all customers to worker 2 is optimal.

Model solution

a) See Figure 1 for a depiction of the queuing network. Using results for the $M/M/1$ queuing system and Little's Law, we have $\bar{T} = \frac{1}{\mu-\lambda} = \frac{1}{1-p}$

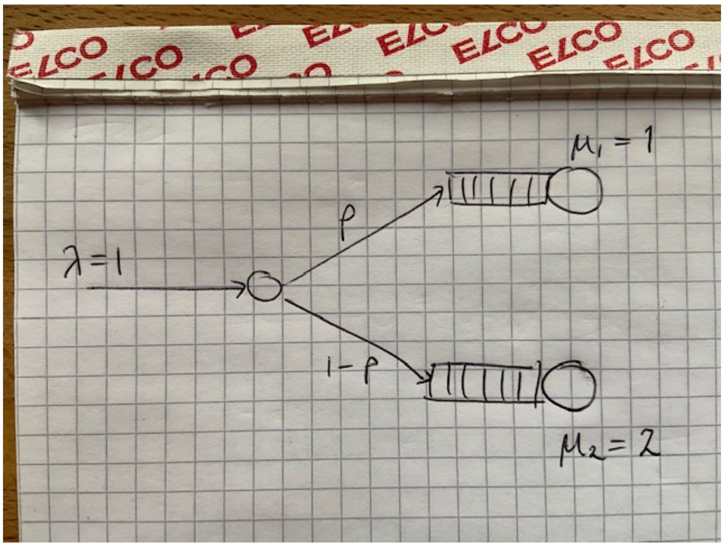


Figure 1: Company Helpline (a)

b) Let T be the total time a customer spends on the call. Using the result from part a, we have

$$\begin{aligned}\mathbb{E}(T) &= p\frac{1}{1-p} + (1-p)\frac{1}{2-(1-p))} \\ &= \frac{p}{1-p} + \frac{1-p}{1+p} \\ &= -2 + \frac{1}{1-p} + \frac{2}{1+p}\end{aligned}$$

To maximise this wrt p we can ignore the constant and differentiate, giving

$$\begin{aligned}\frac{-1}{(1-p)^2} + \frac{2}{(1+p)^2} = 0 \\ \Rightarrow p^2 - 6p + 1 = 0 \\ \Rightarrow p = 3 \pm 2\sqrt{2}\end{aligned}$$

With the minimum at $p^* = 3 - 2\sqrt{2} \approx 0.1716$

c) See Figure 2 for a depiction of the queuing network with inexperienced workers.

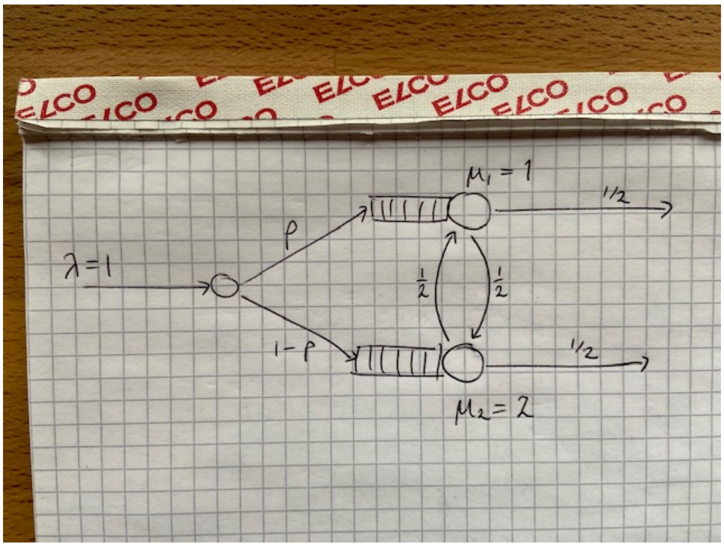


Figure 2: Company Helpline (c)

To calculate the effective arrival rates λ_i, we use the traffic equations.

$$\begin{aligned}\lambda_1 &= p + \frac{1}{2}\lambda_2 \\ \lambda_2 &= (1-p) + \frac{1}{2}\lambda_1 \\ \Rightarrow \lambda_1 &= \frac{2}{3}(1+p), \\ \lambda_2 &= \frac{2}{3}(2-p)\end{aligned}$$

For stability we require that the queues satisfy $\mu_i > \lambda_i$, so we have

$$\begin{aligned}\lambda_1 &= \frac{2}{3}(1+p) < 1 \Rightarrow p < \frac{1}{2} \\ \lambda_2 &= \frac{2}{3}(2-p) < 2 \Rightarrow p > -1\end{aligned}$$

Therefore $0 < p < \frac{1}{2}$.

d) *Solution 1:*

We notice that with $p = 0$ the traffic equations give us the effective arrival rates $\lambda_1 = \frac{2}{3}$ and $\lambda_2 = \frac{4}{3}$. This gives us

$$\begin{aligned}\rho_1 &= \frac{\lambda_1}{\mu_1} = \frac{2}{3} \\ \rho_2 &= \frac{\lambda_2}{\mu_2} = \frac{2}{3}\end{aligned}$$

So effectively both workers have arrival rates directly proportional to their processing rates. This suggests that the workers are balanced even though we are immediately sending all callers to worker 2.

Solution 2:

We notice that with $p = 0$ the traffic equations give us the effective arrival rates $\lambda_1 = \frac{2}{3}$ and $\lambda_2 = \frac{4}{3}$. The effective arrival rate for worker 1 compared to worker 2 is still much higher than the optimum in part (b). So this suggests that the bottleneck is still worker

1, i.e. we are still sending too many callers to worker 1. However we cannot decrease p any further. So $p = 0$ must be optimal.

Solution 3:

Recalling the first exercise from the queueing sheet we can also view the whole call center as a whole system and apply Little's Law to the system as a whole to calculate $\mathbb{E}(T)$. $\mathbb{E}(T) = \frac{\bar{N}}{\lambda}$. We have $\lambda = 1$. And there are $N = \frac{\lambda}{\mu - \lambda}$ in an $M/M/1$ queue in expectation. By linearity of expectation we have

$$\mathbb{E}(T) = \frac{\bar{N}}{1} = \frac{\lambda_1}{\mu_1 - \lambda_1} + \frac{\lambda_2}{\mu_2 - \lambda_2} = \frac{2}{3}\left(\frac{1+p}{1 - \frac{2}{3}(1+p)} + \frac{2-p}{2 - \frac{2}{3}(2-p)}\right)$$

Ignoring constants, this simplifies to

$$\frac{1+p}{1-2p} + \frac{2-p}{2+2p}$$

And simplifies further to

$$\frac{1}{1-2p} + \frac{1}{1+p}$$

Differentiating we get

$$\frac{2}{(1-2p)^2} = \frac{1}{(1+p)^2}$$
$$\Rightarrow p = 2 \pm \frac{3}{\sqrt{2}}$$

Recall for stability we require $p < \frac{1}{2}$. The extrema are outside $\left[0, \frac{1}{2}\right)$, so minimum must be at $p = 0$. All the calls are sent to worker 2.

4 Task Allocation (16 points)

Your company has k servers. Each server can only process one task at a time. Clients send task requests during the day. After receiving a task request, you immediately need to decide whether to accept the task or to refuse it. If a task is accepted, it will be allocated to an available server immediately. The company does not know the duration of a task until it has completed it. The daily company opening hours are T hours and task requests can arrive in the interval $[0, T)$. Each task takes at least 1 hour to process. Any task which is allocated to a server will be processed until it is finished (even after the opening hours, assuming that no task will spill into the next working day). When the company completes 1 task the company receives 1 coin, independent of the task time.

The company employs a natural greedy algorithm to determine whether to accept a task or not: When the company receives a task request i from a client at time t, the company will check if there are any servers available. If so, they will accept the task and allocate it to any available server. If this task takes t_i hours, then the server will be occupied from time t to time $t + t_i$. In other words, other tasks cannot be run on this server from time t to time $t + t_i$. If no servers are available at time t, the company will refuse the task request.

Consider the following example and answer the questions below.

The company has 2 servers. The opening hours of January 28 are $T = 10$ hours, i.e., $t \in [0, 10)$. The following are the task requests from clients:

Task	Arrival Time ($\in [0, T)$)	Duration ($\geqslant$ 1 hour)
1	0	7
2	0.5	4
3	1	3
4	2	2
5	4	5.5
6	4.2	1.5
7	6	3

a) [3] If the company uses the described greedy algorithm to decide which tasks to accept, how many coins can the company earn on January 28?

b) [3] How many coins could they have earned if they had known all tasks and their duration at time 0?

c) [6] The company has k servers, for some given $k > 1$. Is the natural greedy algorithm described above r-competitive for some constant r? If so, what is the minimum possible value of r?

d) [4] The company has k servers, for some given $k > 1$. Is there a deterministic online algorithm with a better competitive ratio than the natural greedy algorithm? Explain your answer.

Model solution

a) The greedy algorithm can earn 3 coins (accept tasks 1, 2 and 7).

b) The optimal offline algorithm can earn 5 coins (accept task 3, 4, 5, 6 and 7).

c) The natural greedy algorithm is n-competitive.

Lower bound: There are k task requests arriving at time 0 and each of them takes n hours. The natural greedy algorithm will accept all of them and any further tasks will not be accepted. After that, at every time $z \in \mathbf{N}, z < n$, k task requests arrive and each of them takes 1 hours. The natural greedy algorithm has utility k and the optimal offline algorithm has utility nk. Therefore, the lower bound of the competitive ratio is n.

Upper bound: Compare the different tasks accepted by the natural greedy algorithm and the optimal offline algorithm, for every accepted task by the natural greedy algorithm, the optimal offline algorithm can accept at most n tasks. Thus, $\frac{Utility_{optimal}}{Utility_{greedy}} \leqslant n$.

d) There is no deterministic online algorithm has a better competitive ratio than the natural greedy algorithm.

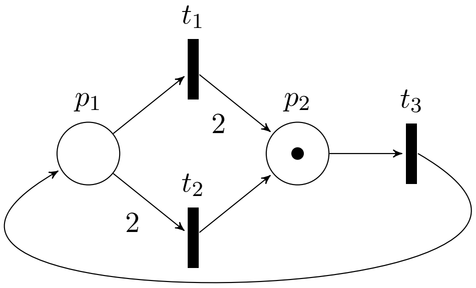
At every time $z \in \mathbf{N}, z < n$, $2k$ task requests are sent to the company. If the company accepts the task i, then task i needs n hours to be completed on a server. If the company refuses the task j, then task j needs 1 hour to be completed on a server. The company can accept at most k tasks. However, the optimal offline algorithm can have kn utility. Therefore, n is the lower bound of the competitive ratio of any deterministic online algorithm. Thus, there is no deterministic online algorithm having a better competitive ratio than the natural greedy algorithm.

5 True or False (6 points)

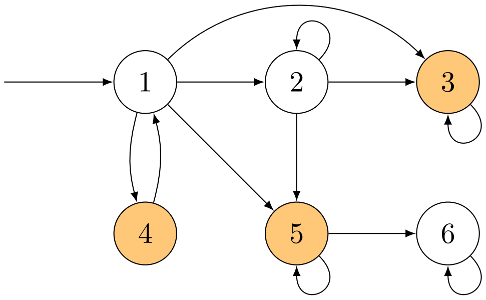
For each of the following statements, assess if it is true or false and tick the corresponding box. No justification is needed.
Every correct answer grants one point. Leaving a statement blank gives 0 point. Every incorrect answer looses one point on the total, with a minimum of 0 point for the whole question.

Notes:
- Petri net **P** is depicted in Figure 3(a).
  Automaton **A** is depicted in Figure 3(b).
- $\llbracket\, p \,\rrbracket$ denotes the set of states which satisfies property p.
  For example for automaton **A**, $\llbracket\, p \,\rrbracket = \{\, 3,\ 4,\ 5\, \}$.

	Statement	**True**	**False**
1	The size of an ordered and reduced Binary Decision Diagram is independent of the variable ordering.		
2	A deterministic finite automaton can always be modeled by a regular (i.e., non-timed) Petri net.		
3	The Petri net **P** is deadlock-free.		
4	In a first case, the marking of the Petri net **P** changes from $M = [0, 1]$ to $M = [0, 2]$ due to the firing of t_1. In a second case, the marking changes from $M = [2, 1]$ to $M = [0, 2]$ due to firing of t_2. The resulting states of the Petri net are different for the two cases.		
5	For automaton **A**, $\llbracket$ EF ((EX $\overline{p}$) `AND` (AG p)) $\rrbracket = \{1, 2, 4\}$.		
6	Automaton **A** satisfies AX $\overline{p}$.		



(a) Petri net **P**



(b) Automaton **A** – 1 is the initial state. 2, 3, 5 and 6 are accepting (or final) states. Property p is true only in states 3, 4, and 5.

Figure 3: Petri net **P** (3(a)) and Automaton **A** (3(b))

6 Binary Decision Diagram (10 points)

a) [6] Given the boolean expression of function f and the ordering of variables $x_2 \prec x_3 \prec x_1 \prec x_4$, construct the reduced ordered binary decision diagram (ROBDD) of f. Merge all equivalent nodes, including the leaves.
Note: Use solid lines for *True* arcs and dashed lines for *False* arcs.

$$f : x_1 \cdot x_2 \cdot (\; \overline{x_3} + x_3 \cdot \overline{x_4} \;) + \overline{x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4}$$

b) [2] Consider the BDD of the function g in Figure 4. Express g as a boolean function.

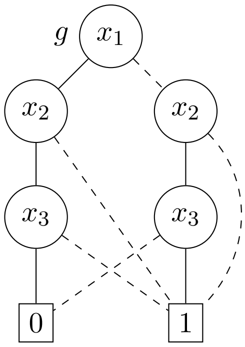


Figure 4: BDD of the boolean function g

c) [2] Simplify the BDD of g (Figure 4) when $x_3 = 0$.

7 Petri nets (24 points)

This question contains 3 independent sub-questions related to Petri nets. Throughout this question, we use the following notations.

- X^t denotes the transpose of vector X.
- $M^t = [p_1, p_2, p_3, p_4]$ and $T^t = [t_1, t_2, t_3, t_4, t_5, t_6]$ are marking and firing vectors of **P** respectively.
  p_i denotes the number of tokens in place i.
  t_i denotes the number of firings of transition i.

Let us first consider the Petri net **P1** in Figure 5.

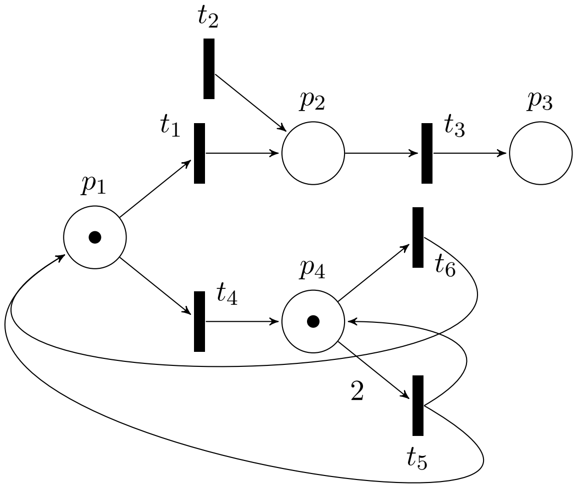


Figure 5: Petri net **P1** – Circles, dots and bars represent places, tokens and transitions, respectively. Arc's weights are marked close by the arc when they are different from 1.

7.1 Reachability `[8 points]`

a) `[2]` Derive the incidence matrix A of the Petri net **P1** from Figure 5.

b) [2] Consider the firing vector $T_S^t = [0, 2, 2, 1, 1, 1]$, where S denotes a firing sequences containing t_2 twice, t_3 twice, and t_4, t_5, t_6 each once.
Use the incidence matrix and the state equation of the Petri net **P1** to compute the marking M_1^t obtained from the initial marking $M_0^t = [1, 0, 0, 1]$ after firing S.

c) [4] Show one example of a firing vector T_P^t and the marking M_P^t obtained from the initial marking $M_0^t = \{1, 0, 0, 1\}$ via the state equation of the Petri net, such that M_P^t is still a valid marking (i.e., containing no negative elements), but there is no valid firing sequence P that has firing vector T_P^t. **Hint:** the firing vector has only one non-zero entry.

7.2 Coverability [8 points]

Let us now consider the Petri net **P2** in Figure 6.

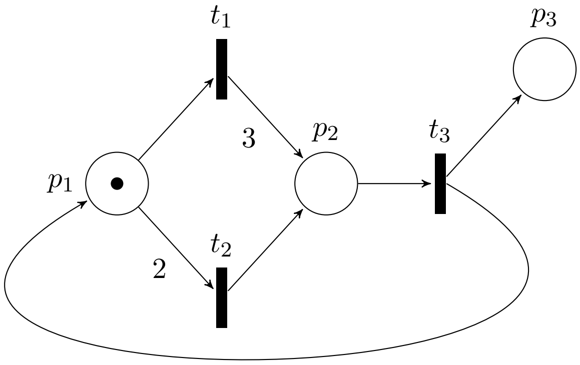


Figure 6: Petri net **P2** – Circles, dots and bars represent places, tokens and transitions, respectively. Arc's weights are marked close to the arc when they are different from 1.

a) [3] Construct the coverability graph of the Petri net **P2**. **Note:** The coverability graph is obtained from the coverability tree by merging nodes with the same marking.

b) [1] From the coverability graph, can you conclude whether or not the marking

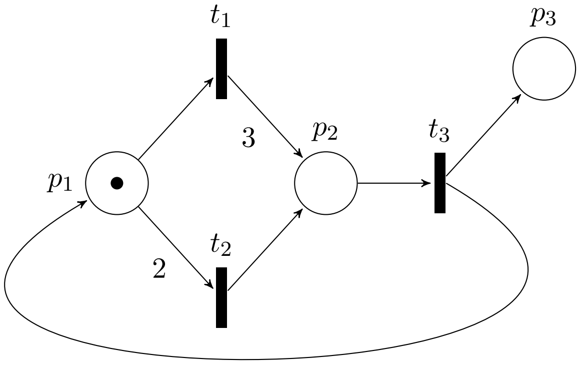
$$M^t = [7, 10, 2]$$

is reachable from the initial marking $M_0^t = [1, 0, 0]$? Justify.

c) [4] Is $M^t = [7, 10, 2]$ reachable from the initial marking $M_0^t = [1, 0, 0]$? Justify.

7.3 Timed Petri Net `[8 points]`

The Petri net **P2** is redrawn below:



But now the transitions are associated with **delays** between their activation and firing:

$$\begin{aligned} d(t_1) &= 3 \\ d(t_2) &= 1 \\ d(t_3) &= 1 \end{aligned}$$

Below is a table for the simulated steps, with columns for number of steps, simulation time τ, firing vector T^τ, current state M^τ, and event list L^τ. Some rows or cells are already given. Simulate the behaviour of the timed Petri net **P2** and fill the the table.

Notes:

- If there are several transitions enabled at the same time, they fire in the ordering of their index, i.e., the smaller index fires first.
- Every firing is a simulated step.

steps	τ	T^τ	M^τ	L^τ
0	0	-	$[1,0,0]$	$(t_1,3)$
1	3	$[1,0,0]$	$[0,3,0]$	$(t_3,4)$
2				
3				
4				
5				