Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

HS 2022 Prof. L. Josipović, Prof. L. Vanbever, Prof. R. Wattenhofer

Discrete Event Systems

Sample Solution

Wednesday, 15th of February 2023, 08:30–10:30.

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.**

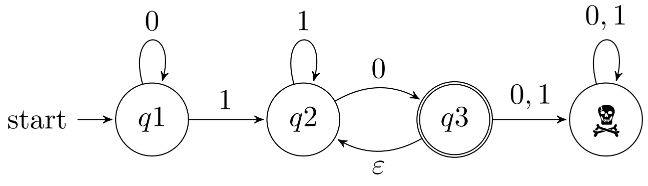
Points

#	Topic	Achieved Points	Max. Points
1	Multiple Choice on Languages & Automata		8
2	Regular Languages		18
3	Context-Free Languages		14
4	Random Bitstrings		20
5	Boxes		20
6	Binary Decision Diagrams		9
7	CTL Model Checking		10
8	Petri Nets		13
9	Time Petri Net		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]`

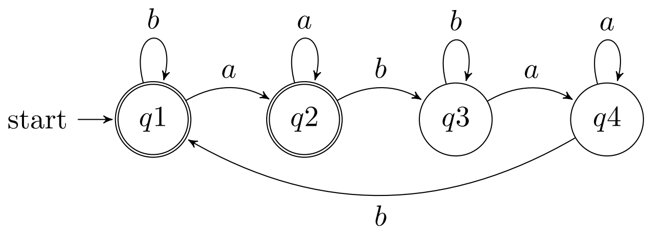
Let $\Sigma = \{0, 1\}$ and consider the automaton A:  start q1 q2 q3 ☠ 0 1 1 0 ε 0, 1 0, 1	**TRUE**	**FALSE**
a) Making q_2 an accepting state does not change the language $L(A)$ recognized by the automaton. *The changed automaton would accept words ending in 1.*	☐	☑
b) The given automaton A is irreducible. *Since this is an NFA, the ☠-state can be safely deleted.*	☐	☑
c) The language $L(A)$ interpreted as unsigned binary numbers describes all positive even numbers. *The changed automaton would newly accept 0^*1^+ as well.*	☑	☐
d) $L(A) = 0^*1^+(0 \cup 10)^*$. *The automaton accepts 10110, but the REX does not.*	☐	☑

1.2 Context-Free Languages `[4 points]`

Let $\Sigma = \{a, (,)\}$ and consider the context-free grammar G: $S \to (\,S\,) \mid T \mid \varepsilon$ $T \to TT \mid a$	**TRUE**	**FALSE**
a) The language $L(G)$ is finite. *$a^+ \subset L(G)$; hence, $L(G)$ is not finite.*	☐	☑
b) There is a GNFA accepting $L(G)$. *$L(G)$ is not regular. It is the language of valid parenthesis expressions; c.f. $L = \{1^n02^n \mid n \geqslant 0\}$ from the lecture with slight modifications in the proof.*	☐	☑
c) The grammar G is ambiguous. *To produce TTT, we need to apply the production $T \to TT$ twice, with two choices which T to double up in the second step. Hence, $T \to^* aaa$ has at least two derivation trees.*	☑	☐
d) L(G) is equivalent to the grammar: $S \to (\,S\,) \mid S\,a \mid \varepsilon$. *This language contains ()a while $L(G)$ does not.*	☐	☑

2 Regular Languages (18 points)

a) Consider the following DFA A with $\Sigma = \{a, b\}$:



(i) `[3]` Describe the language $L(A)$ with your own words.

(ii) `[5]` Consider the following right-linear grammar G:

$$\begin{aligned}
S &\rightarrow A \\
A &\rightarrow bA \mid aB \\
B &\rightarrow aB \mid bC \\
C &\rightarrow bC \mid aD \\
D &\rightarrow aD \mid bA
\end{aligned}$$

Is $L(A) = L(G)$? If yes, argue the equality formally. If no, fix the grammar G and argue the equality afterwards.

b) `[10]` Let $\Sigma = \{0, 1\}$. One of the following languages is regular, while the other is not:

- $L_1 = \{0^k w \mid w \in \Sigma^*, \#_0(w) = k\}$
- $L_2 = \{0^k w \mid w \in \Sigma^*, 1 \leqslant k \leqslant \#_0(w)\}$

State which language is regular, and prove your claim by giving a corresponding DFA, NFA, or REX. Prove that the other language is not regular using the pumping lemma. **Make sure to consider all necessary cases.**

Recall: $\#_0(w)$ *denotes the number of occurrences of the symbol* $0 \in \Sigma$ *in a word* $w \in \Sigma^*$.

Model solution

a) (i) $L(A) = \{w \mid w$ contains the substring "ab" an even number of times $\}$.

(ii) Without changes, $L(G) = \varnothing$ because there is no production that results in only terminal symbols. We can fix the equality $L(G) = L(A)$ by adding the productions $A \rightarrow \varepsilon$ and $B \rightarrow \varepsilon$. To see this, recall that a DFA consists of 5 elements:

i. The grammar G is defined on the same alphabet $\Sigma = \{a, b\}$.
ii. The non-terminal symbols A, B, C, and D represent the states $q_1, ..., q_4$ of A.
iii. S represents the starting state of the automaton which is set to $A \mathrel{\hat{=}} q_1$.
iv. The given production rules for A, B, C, and D define exactly one transition to another state when reading either a or b, respectively. It can be observed that the transitions coincide with the transition function δ of A.
v. The two additional productions $A \rightarrow \varepsilon$ and $B \rightarrow \varepsilon$ represent the accepting states of A.

b)

- We prove that L_1 is not regular using the pumping lemma.
  1. Assume for contradiction that L_1 was regular.
  2. There must exist some p, s.t. any word $s \in L$ with $|s| \geqslant p$ is pumpable.
  3. Choose the string $s = 0^p 1^p 0^p \in L_1$ with length $|s| > p$.
  4. Consider all ways to split $s = xyz$ s.t. $|xy| \leqslant p$ and $|y| \geqslant 1$.
     $\rightarrow$ Hence, $y \in 0^+$.
  5. Observe that $xy^0 z \notin L_1$ – a contradiction to p being a valid pumping length.
  6. Consequently, L_1 cannot be regular. $\square$
- L_2 is regular. To see that, observe that we can assume $k = 1$ to be fixed, as any word $0^k w$ which satisfies the condition for a larger k also satisfies it for the split $0w'$ where $w' = 0^{k-1}w$. Hence, L_1 can be described as $01^*0(0 \cup 1)^*$ and is thus regular.

3 Context-Free Languages (14 points)

A WORD-PDA is a push-down automaton that can recognize either entire words (i.e., a fixed sequence of terminal symbols) on a single transition, or a single symbol from $\{a, b, ..., z, A, ..., Z\}$ by using the special transition-label $'a..Z'$. For example, a WORD-PDA over $\Sigma = \{a, ..., Z\}$ accepting words that start with $'foo'$ or $'bar'$ looks like this:

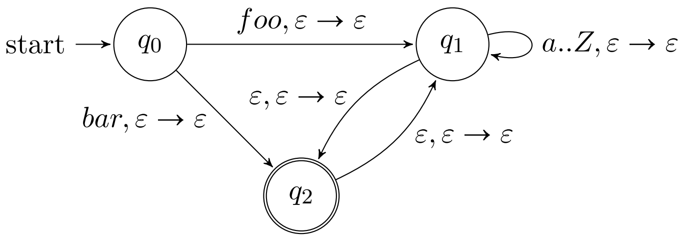
$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[rr, "\substack{foo, \varepsilon \to \varepsilon \\ bar, \varepsilon \to \varepsilon}"] & & q_1 \arrow[loop right, "a..Z, \varepsilon \to \varepsilon"]
\end{tikzcd}$$

We will apply WORD-PDAs to the problem of syntax checking for the Hypertext Markup Language (HTML). We consider the following (simplified) set of tags:

$\underbrace{\texttt{<html>}}_{\text{opening tag}}$ ... $\underbrace{\texttt{</html>}}_{\text{closing tag}}$	`<title>` ... `</title>`	`<h1>` ... `</h1>`
`<head>` ... `</head>`	`<body>` ... `</body>`	`<p>` ... `</p>`

a) [6] Ignoring the real HTML semantics, draw a WORD-PDA using the **minimal number of states** required to check whether all given HTML tags are opened and closed in a correct nesting order. In a correct nesting order, any opened tag must be closed before any previously opened tag is closed. All opened tags must be closed eventually. The WORD-PDA should allow any text between and around the HTML tags.

b) [8] As a variant, consider the 1-WORD-PDA, which is a WORD-PDA that supports at most **one transition** from any state q_i to q_j (including $i = j$), with a single transition-label. Note that a valid 1-WORD-PDA with $\Sigma = \{a, ..., Z\}$ accepting all words starting with $'foo'$ or $'bar'$ requires at least 3 states. For example:


start
q_0
$foo, \varepsilon \to \varepsilon$
q_1
$a..Z, \varepsilon \to \varepsilon$
$bar, \varepsilon \to \varepsilon$
$\varepsilon, \varepsilon \to \varepsilon$
$\varepsilon, \varepsilon \to \varepsilon$
q_2

We will use a 1-WORD-PDA to check the syntax of real HTML documents, for example:

```
<html>
  <head>
    <title>Discrete Event Systems</title>
  </head>
  <body>
    <h1>Regular languages are fun</h1>
    <p>It is no secret that regular languages are fun</p>
  </body>
</html>
```

Draw a 1-WORD-PDA using at most 6 states for the following grammar of HTML:

$S \to \texttt{<}html\texttt{>}\, D\, \texttt{</}html\texttt{>}$ $\qquad\qquad \Sigma = \{\texttt{<}, \texttt{>}, \texttt{/}, 1, a, ..., Z\}$

$D \to \texttt{<}head\texttt{>}\, H\, \texttt{</}head\texttt{>}\, \texttt{<}body\texttt{>}\, B\, \texttt{</}body\texttt{>}$

$H \to \texttt{<}title\texttt{>}\, T\, \texttt{</}title\texttt{>}$

$B \to \texttt{<}h1\texttt{>}\, T\, \texttt{</}h1\texttt{>}\, P\, B \mid \varepsilon$

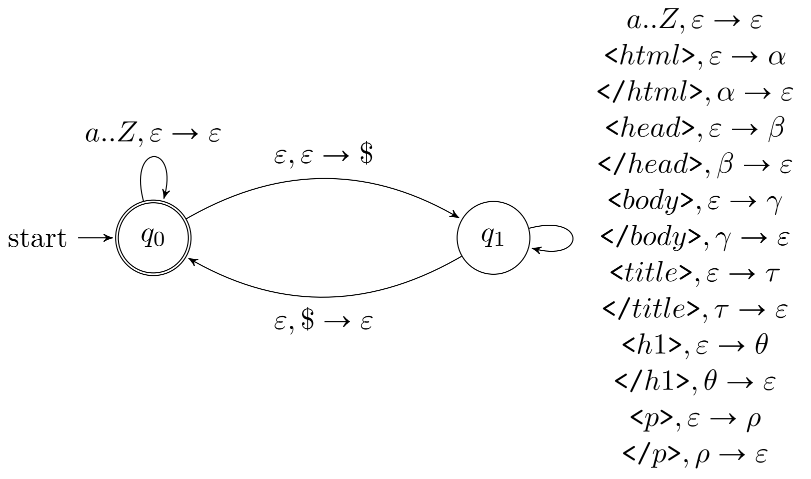
$P \to \texttt{<}p\texttt{>}\, T\, \texttt{</}p\texttt{>}$

$T \to T\,T \mid a \mid b \mid ... \mid Z$

(You will be awarded up to 4 points if your solution uses at most 10 states.)

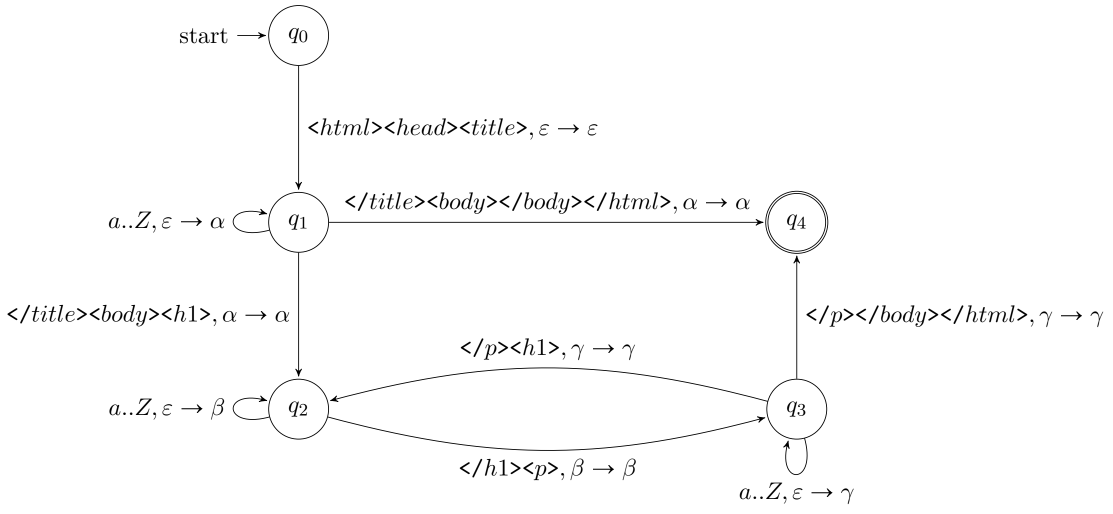
Model solution

a) A minimal WORD-PDA checking the correctness of nesting orders must have at least two states as it cannot recognize all the (infinitely many) combinations with finitely many transitions. A solution with two states for $\Sigma = \{<, >, /, a, ..., Z\}$ looks as follows:



$a..Z, \varepsilon \rightarrow \varepsilon$
start
q_0
$\varepsilon, \varepsilon \rightarrow \$$
q_1
$\varepsilon, \$ \rightarrow \varepsilon$
$a..Z, \varepsilon \rightarrow \varepsilon$
<*html*>, $\varepsilon \rightarrow \alpha$
</*html*>, $\alpha \rightarrow \varepsilon$
<*head*>, $\varepsilon \rightarrow \beta$
</*head*>, $\beta \rightarrow \varepsilon$
<*body*>, $\varepsilon \rightarrow \gamma$
</*body*>, $\gamma \rightarrow \varepsilon$
<*title*>, $\varepsilon \rightarrow \tau$
</*title*>, $\tau \rightarrow \varepsilon$
<*h*1>, $\varepsilon \rightarrow \theta$
</*h*1>, $\theta \rightarrow \varepsilon$
<*p*>, $\varepsilon \rightarrow \rho$
</*p*>, $\rho \rightarrow \varepsilon$

b) A 1-WORD-PDA with five states that recognizes the language looks as follows:



start
q_0
<*html*><*head*><*title*>, $\varepsilon \rightarrow \varepsilon$
$a..Z, \varepsilon \rightarrow \alpha$
q_1
</*title*><*body*></*body*></*html*>, $\alpha \rightarrow \alpha$
q_4
</*title*><*body*><*h*1>, $\alpha \rightarrow \alpha$
</*p*></*body*></*html*>, $\gamma \rightarrow \gamma$
</*p*><*h*1>, $\gamma \rightarrow \gamma$
$a..Z, \varepsilon \rightarrow \beta$
q_2
q_3
</*h*1><*p*>, $\beta \rightarrow \beta$
$a..Z, \varepsilon \rightarrow \gamma$

Note how we utilize the stack to ensure that the page title, headings and paragraphs cannot be empty by requiring to read the symbols α, β, or γ on any outgoing edge from the states q_1, q_2, and q_3, respectively.

4 Random Bitstrings (20 points)

You are given two random binary strings a and b, both of length 3, e.g. a = "011". In each round an index i, where $1 \leqslant i \leqslant 3$, is chosen uniformly at random. If the i^{th} bits of strings a and b are different, we flip the i^{th} bit of string a.

a) `[6]` What is the expected time (number of rounds) until string a equals string b for the first time? Show all your work.

Let us now change the game a bit: You are given two random binary strings c and d of length 3. In each round an index i, where $1 \leqslant i \leqslant 3$, is chosen uniformly at random and the i^{th} bit of string c is flipped (regardless of the i^{th}-bit of string d). You stop when $c = d$.

b) `[3]` Model the game as a Markov Chain.

c) [7] What is the expected time until $c = d$? Show all your work.

Finally, you are again given two random binary strings e and f of length 3. In each round, two indices i and j, where $1 \leqslant i, j \leqslant 3$, are chosen uniformly at random. You flip both the i^{th} bit of string e and the j^{th} bit of string f.

d) [4] What is the expected time until $e = f$? Show all your work.

Model solution

a) The expected number of rounds of the game is

$$\mathbb{E}[\text{number rounds game}] = \sum_{i=1}^{3} \mathbb{P}[\text{probabilitys differ in } i \text{ bits}] \cdot X_i,$$

where X_i is the expected number of rounds of the game if they initially differ in i bits. The probability that the initial strings differ in i positions, where $0 \leqslant i \leqslant 3$, is $\binom{3}{i}\frac{1}{2^3}$. To go from a state in which a and b differ in i bits, where $1 \leqslant i \leqslant 3$, to a state where they differ in $i-1$ bits, one of the bits in which they differ needs to be selected. This happens with probability $\frac{i}{3}$. Thus, in expectation, we remain $\frac{3}{i}$ rounds in the state where the two strings differ by i bits, before going to a state where they differ in $i-1$ bits. Thus the total amount of rounds to go from i different bits to 0 different bits is $\sum_{j=1}^{i} \frac{3}{j}$. Thus

$$\mathbb{E}[\text{number rounds game}] = \sum_{i=1}^{3} \binom{3}{i} \frac{1}{2^3} \cdot \sum_{j=1}^{i} \frac{3}{j} = \frac{7}{2}$$

b) The Markov chain consists of 4 states, indicating the number of bits that differ between two strings. The transitions correspond to the probability that a bit in which the two strings differ respectively equal is selected.

$$\begin{tikzcd}
3 \arrow[r, bend left, "1"] & 2 \arrow[l, bend left, "\frac{1}{3}"] \arrow[r, bend left, "\frac{2}{3}"] & 1 \arrow[l, bend left, "\frac{2}{3}"] \arrow[r, bend left, "\frac{1}{3}"] & 0
\end{tikzcd}$$

Figure 1: Markov chain for the game in part a).

c) For the expected value, we need to determine two things: 1) the initial distribution, and 2) the hitting time from any state to the state that represents a difference of 0 bits between the two strings.

The probability that the two states are equal or complement is $\frac{1}{2^3} = \frac{1}{8}$. The probability that they differ in 1 or 2 bits is $\binom{3}{1}\frac{1}{2^3} = \frac{3}{8}$. Therefore, the starting distribution is $\sigma = (\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8})$.

To compute the hitting times we need to solve the following linear system of equations:

$$h_3 = 1 + h_2 \qquad h_2 = 1 + \frac{2}{3}h_1 + \frac{1}{3}h_3 \qquad h_1 = 1 + \frac{2}{3}h_2$$

To solve this we can substitute the first and third equations into the second which gives

$$h_2 = 1 + \frac{2}{3}\left(1 + \frac{2}{3}h_2\right) + \frac{1}{3}(1 + h_2) \qquad \iff \qquad h_2 = 9$$

Substituting this into the first and third equation gives $h_1 = 7$ and $h_3 = 10$ This implies that the expected number of rounds until strings c and d are equal is

$$\frac{1}{8} \cdot 10 + \frac{3}{8} \cdot 9 + \frac{3}{8} \cdot 7 = 7.25.$$

d) In any round, the number of bits that differ between the two strings either remains the same or changes by two. Therefore, if we start with 1 or 3 different bits, it is not possible to reach equality. Thus the expected number of steps until the two strings are equal is unbounded.

5 Boxes (20 points)

You play the following game consisting of n rounds: In each round you receive a single box with a number written on it that indicates how much money is inside the box. Unfortunately, you only have a limited amount of space available. More precisely, you have enough space to keep a single box in storage, we call this the stored box from now on. Therefore, at each step i you have to decide if you want to stick with your currently stored box or exchange it with the box that you receive in the i^{th} round. If you decide to discard a box, you will never have access to it again.
At the end of the n rounds you will receive the money inside the stored box. Your goal is to maximize the amount of money you can win. The number of rounds n is known to you before the game.

Since we maximize the amount of money, we are interested in "gain" instead of "cost". Recall the definition: An online algorithm A is strictly r-competitive if for all finite input sequences I

$$cost_A(I) \leqslant r \cdot cost_{opt}(I), \qquad \text{or} \qquad r \cdot gain_A(I) \geqslant gain_{opt}(I).$$

a) [4] What is the best deterministic online algorithm and what is its strictly competitive ratio? Prove your claim.

We change the game slightly: Only two boxes will arrive, so $n = 2$. However, this time one of the boxes is a "trap", in the sense that it does not contain any money (no matter what is written on it). The optimal offline algorithm knows which box is the trap.

b) [3] Is there a deterministic online algorithm that is strictly constant competitive? Prove your claim.

c) `[4]` Can you design a randomized online algorithm that is strictly 2-competitive? Prove your claim.

d) `[5]` Show that there is no randomized online algorithm that is better than strictly 2-competitive. Prove your claim.

Now you will receive a lot of boxes, so n will be very large. Again, one of the boxes is a trap that does not contain any money. But this time you are guaranteed that all boxes have “10” written on them.

e) `[4]` Can you design a randomized online algorithm with best possible strictly competitive ratio as n goes to infinity? Prove your claim.

Model solution

a) The algorithm always picks the box with the highest value and puts it as the stored box. So if a new box has a larger value written on it, it will be exchanged with the stored box. This strategy is strictly 1-competitive because the gain corresponds to the largest of the n boxes. Furthermore, this is best possible as no algorithm can get more money than written on any of the boxes.

b) There is no deterministic algorithm that is strictly constant competitive. Let's look at the sequence of two boxes a, b. Wlog. we can assume that the algorithm always pick box a to be stored first, so we can choose between keeping the box with a or b written on it. Because the algorithm is deterministic we know in advance which of the two boxes will be chosen. So an adversary can remove the money from the chosen box and the deterministic algorithm will always have gain 0 whereas the optimal algorithm could always choose the box containing the money. Therefore, no deterministic algorithm can be strictly (constant) competitive.

c) We keep the first box and then compare it to the second box that arrives. With probability $\frac{1}{2}$ we exchange the boxes (no matter what is written on them). The strategy is 2-competitive in expectation. Let x be the value written on the box which contains money. Clearly, the optimal player could gain at most x so $x \geqslant gain_{OPT}(I)$ holds. Using our proposed algorithm, we pick the box containing x money and the box with no money with equal probability. Our expected gain is $\frac{1}{2}x + \frac{1}{2}0$. Therefore, $r \cdot gain_A(I) = 2 \cdot \frac{1}{2}x = x \geqslant gain_{OPT}(I)$ holds and the expected competitive ratio is 2.

d) No, use the Yao principle with the following input distribution consisting of two inputs which are each chosen with probability $\frac{1}{2}$. The first box a has value v_a, the second box b has value v_b. In the first input box a is the trap, in the second input box b is the trap. Note that any deterministic algorithm will always choose either a or b for both inputs. Therefore, its total gain will be either v_a or v_b, compared to the optimal offline algorithm who will have a gain of $v_a + v_b$. Therefore the competitive ratio is either $\frac{v_a+v_b}{v_a}$ or $\frac{v_a+v_b}{v_b}$. By choosing $v_a = v_b$, we get that $\frac{v_a+v_b}{v_a} = \frac{v_a+v_b}{v_b} = 2$ is a lower bound on the competitive ratio for all deterministic algorithms on the chosen input distribution. By Yao it then follows that there is no randomized algorithm which is better than 2 competitive.

e) Choosing one of the boxes in the beginning with equal probability gets you a $\frac{n}{n-1}$-competitive algorithm. All the boxes have the same amount of money in them and the probability that you choose one of them is $\frac{n-1}{n}$. Because all boxes with money contain the same amount of money we have: $10 \geqslant gain_{OPT}(I)$ and our expected gain is $\frac{n-1}{n}10 + \frac{1}{n}0$. We put this together to match the definition of the competitive ratio with $r = \frac{n}{n-1}$: $r \cdot gain_A(I) \geqslant \frac{n}{n-1} \cdot \frac{n-1}{n}10 \geqslant gain_{OPT}(I)$. As n goes to infinity the ratio tends towards one, i.e. will be as good as the optimum algorithm.

6 Binary Decision Diagrams (9 points)

The following truth table represents a Boolean function $f(a, b, c)$.

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 1: Truth table.

a) [2] Given the Boolean expression of f in the table above and the ordering of variables $a \rightarrow b \rightarrow c$ (i.e., a is the top node), construct the **non-reduced** (i.e., with 7 nodes and 8 leaves) ordered binary decision diagram (OBDD) of f.
Note: In all questions, use solid lines for *True* arcs and dashed lines for *False* arcs.

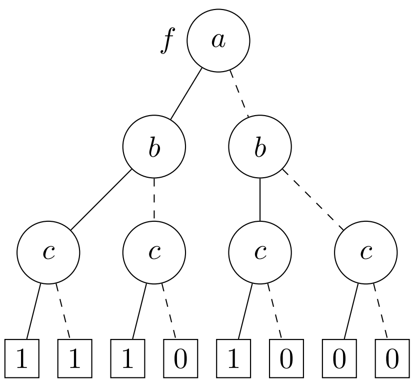
b) `[2]` Create the reduced ordered binary decision diagram (ROBDD) of f by merging equivalent nodes and leaves of the OBDD obtained in the previous question.

c) `[2]` Would another variable ordering produce a smaller ROBDD? If yes, show the ROBDD for this order. If no, elaborate on your answer.

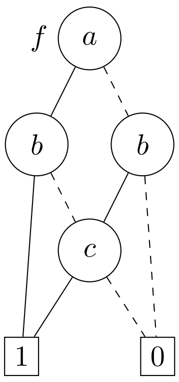
d) `[3]` Build a BDD for $g = \forall b : f$. Use the same variable order as in question 6-**a)**.

Model solution

a) OBDD of the Boolean function in Table 1.



b) ROBDD of f



c) No, the ROBDD will not become smaller: we apply logic simplification to f

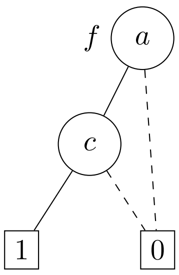
$$f(a,b,c) = a \cdot b + b \cdot c + a \cdot c, \tag{1}$$

which is fully symmetrical, i.e., switching the labels preserves the expression.

d) We apply the definition of the universal quantifier

$$\forall b : f(a,b,c) := f(a,0,c) \cdot f(a,1,c), \tag{2}$$

$$\forall b : f(a,b,c) = (a \cdot c) \cdot (a + c) = a \cdot c. \tag{3}$$



7 CTL Model Checking (10 points)

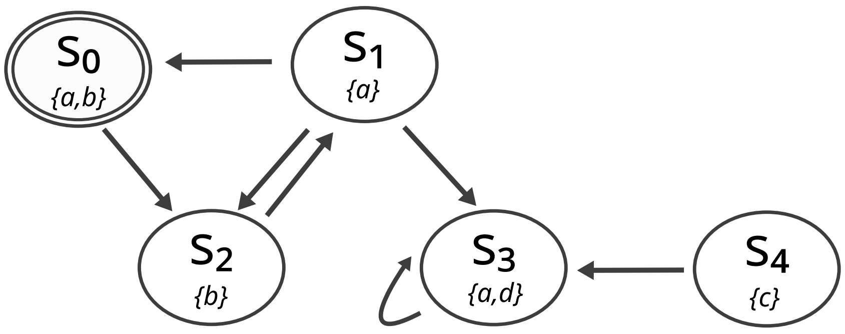


Figure 2: State transition diagram.

Consider a state transition diagram illustrated in Figure 2, with the *set of atomic propositions* $\{a, b, c, d\}$, the *set of states* $\{s_0, s_1, s_2, s_3, s_4\}$, and the *set of initial states* $\{s_0\}$. The transition relations are given as edges between the states; each state is labeled with a set of atomic propositions that are true at the state.

a) [2] Describe the meaning of each of the following CTL formulas over the set of atomic propositions in plain language.

- $\mathbf{AG}\ (a \vee b)$.

- $\mathbf{A}\ [(a \vee b)\ \mathbf{U}\ d]$.

b) [2] For each of the formulas above, give the **set of states** that satisfies the formula.

c) [6] Formulate an iterative procedure to find the set of states that satisfy the CTL formula **AF EG** d. Indicate the initial set of states and all visited sets of states until you reach a fixed point.

Note: For a given set of states Q, we define the predecessor function $P(Q)$ as a function that returns the set of all direct predecessor states of set Q; we define the successor function $S(Q)$ as a function that returns the set of all direct successor states of set Q.

Model solution

a) Translate CTL formula to plain language:

- $\mathbf{AG}\ (a \vee b)$: In all reachable states, either a or b holds.
- $\mathbf{A}\ [(a \vee b)\ \mathbf{U}\ d]$: For all paths, either a or b has to hold until d holds.

b) Set of states that satisfy the CTL formula:

- $\mathbf{AG}\ (a \vee b)$: $\{s_0, s_1, s_2, s_3\}$.
- $\mathbf{A}\ [(a \vee b)\ \mathbf{U}\ d]$: $\{s_0, s_1, s_2, s_3\}$.

c) CTL Model Checking:

We first compute the set of states that satisfy $\mathbf{EG}\ d$. We start from the set of states that satisfy d:

$$Q_0 = \{s_3\}. \tag{4}$$

We then compute $P(Q_0)$, the set of all direct predecessors of Q_0, and compute the intersection between Q_0 and $P(Q_0)$:

$$Q_1 = Q_0 \cap P(Q_0) = \{s_3\} \cap \{s_3, s_1, s_4\} = \{s_3\}, \tag{5}$$

and here we have detected that a fix-point is reached ($Q_1 \equiv Q_0$), therefore the set of states that satisfy $\mathbf{EG}\ d$ is $\{s_3\}$. We define a new atomic proposition ϕ as the set of states that satify $\mathbf{EG}\ d$.

Now we compute the set of states that satisfy $\mathbf{AF}\ \phi$. We use the relation $\mathbf{AF}\ \phi = \neg\mathbf{EG}\ \neg\phi$. We start the procedure by identifying the set of states that satisfy $\neg\phi$, which is given as

$$Q_0 = \{s_0, s_1, s_2, s_4\}. \tag{6}$$

We then compute $P(Q_0)$, the set of all direct predecessors of Q_0, and compute the intersection between Q_0 and $P(Q_0)$, which is given as

$$Q_1 = Q_0 \cap P(Q_0) = \{s_0, s_1, s_2, s_4\} \cap \{s_0, s_1, s_2\} = \{s_0, s_1, s_2\}. \tag{7}$$

Since a fixed-point is not reached, we repeat the process to obtain Q_2:

$$Q_2 = Q_1 \cap P(Q_1) = \{s_0, s_1, s_2\} \cap \{s_0, s_1, s_2\} = \{s_0, s_1, s_2\}, \tag{8}$$

and here we have detected that a fix-point is reached ($Q_2 \equiv Q_1$). Therefore, the set of states that satisfy $\mathbf{EG}\ \neg\phi$ is $\{s_0, s_1, s_2\}$. We then compute the complement of this set, which is given as

$$\mathbf{AF}\ \mathbf{EG}\ d = \mathbf{AF}\ \phi = \neg\mathbf{EG}\ \neg\phi = \{s_3, s_4\}. \tag{9}$$

8 Petri Nets (13 points)

Liveness and Capacity

a) [2] Consider the Petri net in Figure 3. Determine the highest liveness level of all transitions (t_1 to t_4) and concisely elaborate on your answers.

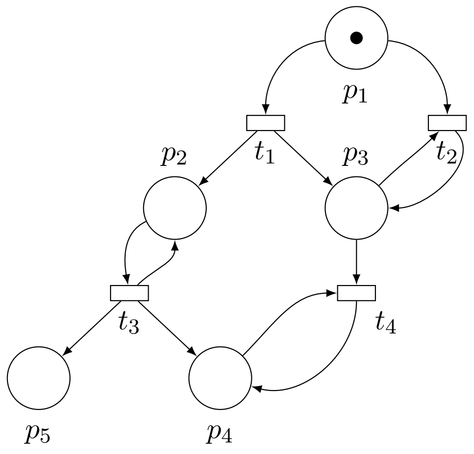


Figure 3: Petri net for liveness analysis.

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 L1 live for every marking that is reachable from M_0.

L_{j+1} liveness implies L_j liveness.

b) [1] Figure 4 repeats the Petri net of Figure 3. Modify the Petri net in the figure to create a capacity constraint of 2 tokens in place p_5.

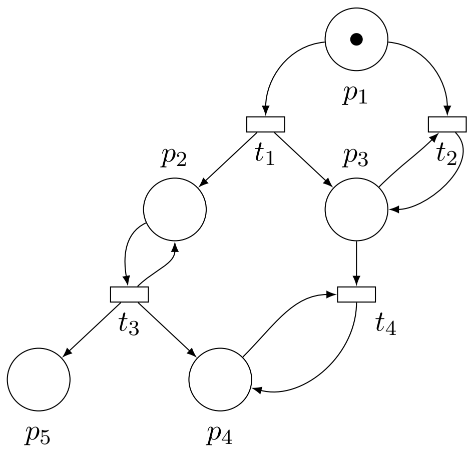


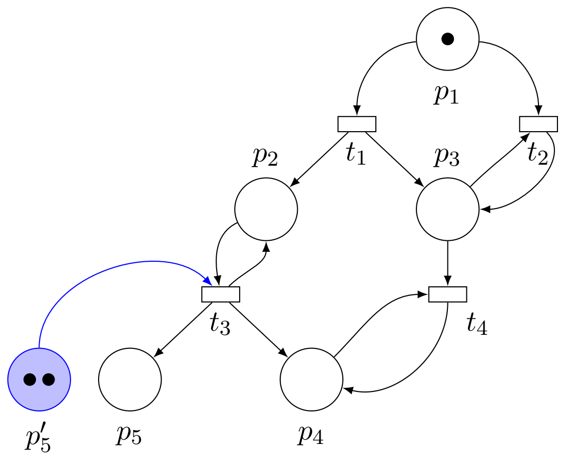
Figure 4: Petri net for capacity constraint.

c) [2] Determine the liveness level of all transitions (t_1 to t_4) in the modified Petri net (i.e., with a capacity constraint of 2 tokens in place p_5) and concisely elaborate on your answers.

Model solution

a)
- t_1 is L_1-live.
- t_2 is dead.
- t_3 is L_4 live.
- t_4 is L_1 live.

b) We add an extra place p_5' with two initial tokens and an arc from p_5' to t_3.



c)
- t_1 is L_1-live.
- t_2 is dead.
- t_3 is L_1 live.
- t_4 is L_1 live.

Reachability

Consider a Petri net with four places, p_1 to p_4, and three transitions, t_1 to t_3, characterized by the following initial marking M_0 and incidence matrix A.

$$M_0 = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad A = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix} \tag{10}$$

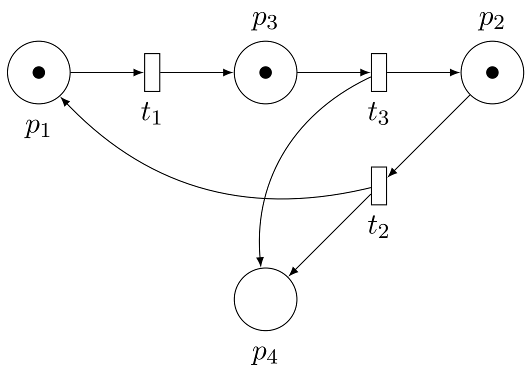
a) [2] Draw the Petri net described by M_0 and A. Clearly label all places and transitions. Indicate the initial token marking.
Note: In a marking M, element m_i denotes the number of tokens in place p_i. In the incidence matrix A, element $A_{i,j}$ describes the "gain" of tokens at place p_i when transition t_j fires.

b) [3] Is the marking $M_1 = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 4 \end{bmatrix}$ reachable from M_0? Justify your answer.

c) [3] Is the marking $M_2 = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 4 \end{bmatrix}$ reachable from M_0? Justify your answer.

Model solution

a) The Petri net described by M_0.



b) The marking is reachable. We prove reachability by giving a firing sequence. The following two sequences are all valid solutions:

- $\{t_2, t_3, t_2, t_1, t_3\}$.
- $\{t_2, t_3, t_1, t_2, t_3\}$.

c) The marking is *not* reachable. We have two approaches:

- We can identify a place invariant: $m(p_1) + m(p2) + m(p3) = 3$. And the given marking does not satisfy the invariant.
- The system of linear equations given by $M_0 + A \cdot x = M_2$ has no solution.

9 Time Petri Net (8 points)

Consider the Petri net in Figure 5.

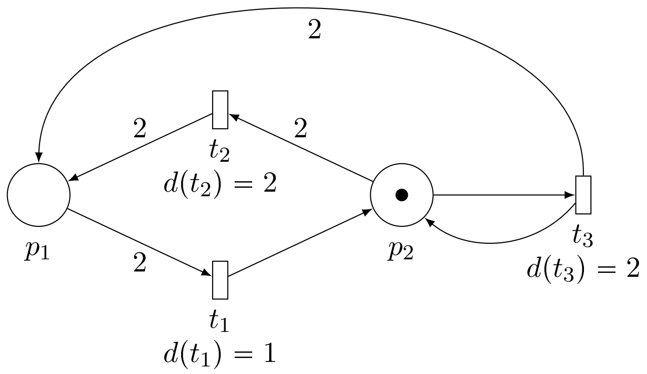


Figure 5: Time Petri net at simulation step 0 ($\tau = 0$).

The transitions are associated with the following delays between their activation and firing: $d(t_1) = 1$, $d(t_2) = 2$, $d(t_3) = 2$.

a) `[8]` 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. The first two simulation steps are already indicated in the table.

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	-	$[0, 1]$	(t_3, 2)
1	2	t_3	$[2, 1]$	(t_1, 3), (t_3, 4)
2				
3				
4				
5				

Model solution

a) Simulation of the Petri net:

step	τ	t_{fired}	M^τ	L^τ
0	0	-	$[0, 1]$	$(t_3,\, 2)$
1	2	t_3	$[2, 1]$	$(t_1,\, 3)$, $(t_3,\, 4)$
2	3	t_1	$[0, 2]$	$(t_3,\, 4)$, $(t_2,\, 5)$
3	4	t_3	$[2, 2]$	$(t_1,\, 5)$, $(t_3,\, 6)$, $(t_2,\, 6)$
4	5	t_1	$[0, 3]$	$(t_3,\, 6)$, $(t_2,\, 6)$
5	6	t_2	$[2, 1]$	$(t_3,\, 8)$, $(t_1,\, 7)$