Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and Networks Laboratory
NSG & DISCO

HS 2021 Dr. R. Jacob, Prof. L. Vanbever, Prof. R. Wattenhofer

Discrete Event Systems

Sample Solution

Thursday, 27th of January 2022, 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.**

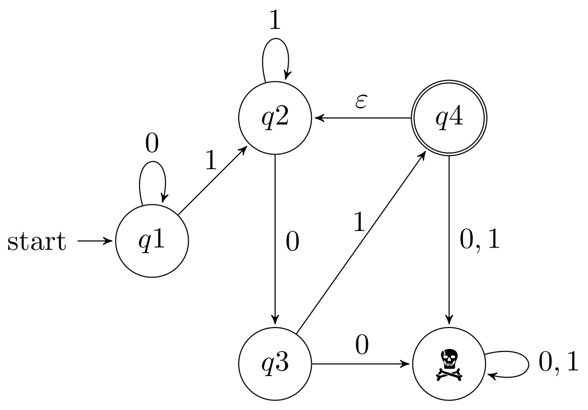
Points

#	**Topic**	**Achieved Points**	**Max. Points**
1	Multiple Choice on Languages & Automata		8
2	Regular & Context-Free Languages		18
3	Tandem-Pumping Lemma		14
4	Queueing Networks		22
5	Card Game		18
6	Multiple Choice on Model Checking		6
7	Binary Decision Diagram		10
8	Computation Tree Logic		8
9	Petri Nets		16
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: 	**TRUE**	**FALSE**
a) The given automaton A is a DFA. *There is an ε-transition.*	☐	☑
b) Making q_2 an accepting state does not change the language $L(A)$ recognized by the automaton. *The changed automaton would newly accept 0^*1^+ as well.*	☐	☑
c) $L(A) = (10)^+1 \cup 0^*1(1^*01)^+$. $(10)^+1 \quad \subset \quad 0^*1(1^*01)^+ \quad = L(A)$.	☑	☐
d) There are always more 1s than 0s. *The automaton starts with a self-loop that accepts an arbitrary number of leading 0s.*	☐	☑

1.2 Context-Free Languages `[4 points]`

Let $\Sigma = \{0,1\}$ and consider the grammar G: $S_0 \to S_0\,0 \mid S_1\,1 \mid \varepsilon$ $S_1 \to S_2\,0 \mid S_0\,1$ $S_2 \to S_1\,0 \mid S_2\,1$	**TRUE**	**FALSE**
a) The grammar G is given in Chomsky Normal Form (CNF). *All terminal symbols must be produced from separate productions in CNF.*	☐	☑
b) The language $L(G)$ is regular. *G is a left-linear grammar. (Alternative: $L(G)$ equals an automaton to check whether the input is divisible by 3.)*	☑	☐
c) The grammar G is ambiguous. *There is always exactly one transition for any terminal symbol read from the input.*	☐	☑
d) $01001 \in L$. $S_0 \Rightarrow S_1\,1 \Rightarrow S_2\,0\,1 \Rightarrow S_1\,0\,0\,1 \Rightarrow S_0\,1\,0\,0\,1 \Rightarrow S_0\,0\,1\,0\,0\,1 \Rightarrow$ 01001. *(Alternative: $01001_{(2)} = 9_{(10)}$ which is divisible by 3.)*	☑	☐

2 Regular & Context-Free Languages (18 points)

a) In this task, we consider the art of Integer Pruning, that is, just like one can cut trees into nice shapes, we now cut numbers to look nice by cropping selected digits from them. More specifically, we like numbers that are divisible by 25, but we **do not like the number 0.**

1234567 → **12**~~34~~**5**~~67~~ = 125

Your task is to find out whether a number can be pruned to a "nice number" (i.e. divisible by 25 and greater than 0) by cropping digits from it. As for trees, we should not cut away the entire number.

More formally, consider the language

$$L = \{n \in \mathbb{N} \mid n \text{ can be pruned to a “nice number”}\}$$

For example, ***1***~~*2*~~***35***~~*71*~~***0*** *and* ***7***~~*6*~~***5***~~*4321*~~ $\in L$, *but* 54321 $\notin L$.

(i) [2] Find two 5-digit numbers that belong to L.

(ii) [8] Draw a non-deterministic automaton recognizing L using at most 5 states.
(You will be awarded up to 7 points if your solution uses at most 7 states.)

b) [8] Draw a non-deterministic PDA using at most 8 states that recognizes the language:

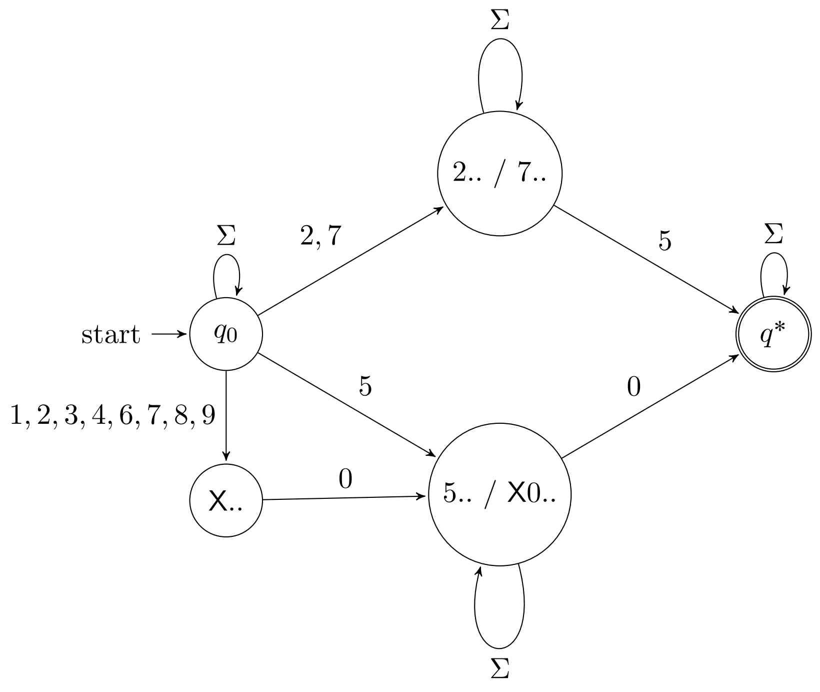
$$L = \{x\#y\#z \mid x \in \{0,1\}^+,\ y, z \in \{0,1\}^*,\ |x| \neq |z|\}.$$

For example, $0\#1\#11$ *and* $00\#\#000 \in L$*, but* $0\#\#1 \notin L$*.*

Model solution

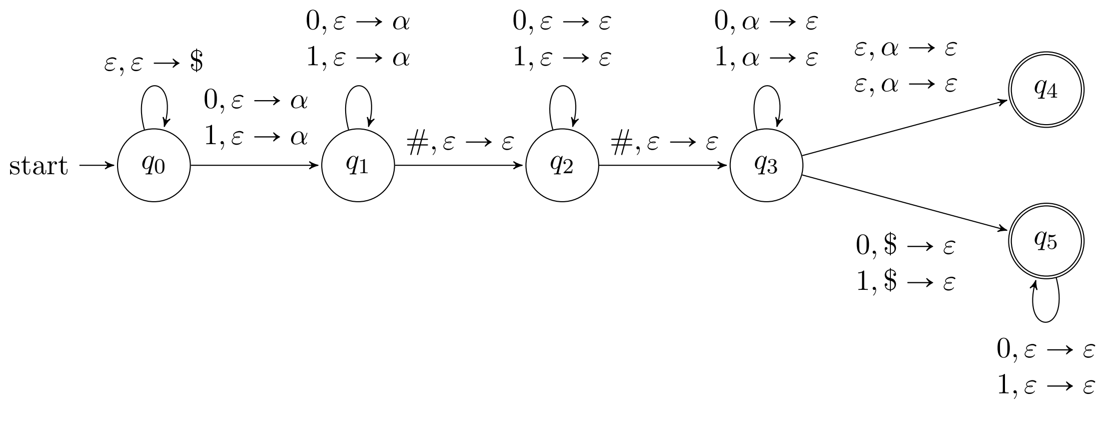
a) (i) Consider 21115 →**2**~~111~~**5** = 25 and 17151 → ~~1~~**7**~~1~~**5**~~1~~ = 75.

(ii) All numbers accepted must either contain the digits of 25, 50, 75 or some non-zero digit X and two zeros. (Note that X = 5 is already covered.)



Remark: As it seems instructive, we allow the short notation of Σ on a transition to indicate that any character from Σ would match here. This is ***not*** *generally valid syntax for an automaton.*

b) A PDA recognizing $L = \{x\#y\#z \mid x \in \{0,1\}^+,\ y, z \in \{0,1\}^*,\ |x| \neq |z|\}$ could look like this:



start
q_0
$\varepsilon, \varepsilon \to \$$
$0, \varepsilon \to \alpha$
$1, \varepsilon \to \alpha$
q_1
$0, \varepsilon \to \alpha$
$1, \varepsilon \to \alpha$
$\#, \varepsilon \to \varepsilon$
q_2
$0, \varepsilon \to \varepsilon$
$1, \varepsilon \to \varepsilon$
$\#, \varepsilon \to \varepsilon$
q_3
$0, \alpha \to \varepsilon$
$1, \alpha \to \varepsilon$
$\varepsilon, \alpha \to \varepsilon$
$\varepsilon, \alpha \to \varepsilon$
q_4
$0, \$ \to \varepsilon$
$1, \$ \to \varepsilon$
q_5
$0, \varepsilon \to \varepsilon$
$1, \varepsilon \to \varepsilon$

The automaton halts in q_4 when $|x| > |z|$ and in q_5 when $|x| < |z|$. Note that we need to check that x cannot be empty, as $x \in \{0,1\}^+$. In return, we can save one state by using a self-loop on q_0 to introduce a \$ sign on the stack only if necessary. The automaton may never reach q_5 if it does not put a \$ sign on the stack initially, and to check whether $|x| > |z|$, we would never read the \$ sign anyway.

3 Tandem-Pumping Lemma (14 points)

Consider the language

$$L = \left\{ a\#b\#c \;\middle|\; a, b, c \in \{0,1\}^*,\ c = 2a,\ \#_0(b) = \#_0(c) \right\}$$

for unsigned binary numbers a, b, and c. For example, $0\#10\#0 \in L$ and $1\#00\#010 \in L$.

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

a) [4] Show that $w = 1^p\#0\#1^p0$ is tandem-pumpable in L.

Hint: Split up $w = uvxyz$ *such that* $x = \#0\#$.

b) [10] Use the tandem-pumping lemma to show that L is not context-free.

Hint: Choose a string $w = a\#b\#c$ *where* $1 \notin b$, *i.e.* $b \in 0^*$.

Model solution

a) $w = 1^p\#0\#1^p0$ is tandem-pumpable for the split $w = uvxyz$ where $u = 1^{p-1}$, $v = 1$, $x = \#0\#$, $y = 1$, and $z = 1^{p-1}0$:

- $w \in L$, because "1^p0" $= 2 \cdot$ "1^p" and $\#_0(b) = 1 = \#_0(c)$.
- $uv^0xy^0z = 1^{p-1}\#0\#1^{p-1}0$, which is in L.
  (i.e. removing v and y from w does not break any of the language's rules)
- v and y are part of a's and c's leading 1s, respectively. As $v = y = 1$, both numbers are modified identically, while c's trailing 0 ensures that $c = 2a$ remains true.
- v and y do not contain any 0s, so $\#_0(b) = \#_0(c)$ is preserved.

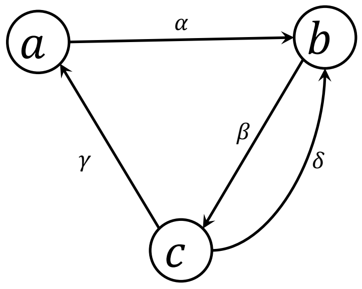
b) We prove that L is not context-free using the tandem-pumping lemma.

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 string $w = 10^{p-1}\#0^p\#10^p \in L$ with length $|w| > p$.
4. Consider all ways to split $w = uvxyz$ s.t. $|vxy| \leqslant p$ and $|vy| \geqslant 1$.
   - First, we observe that if the vxy part was completely part of a, b, or c (for $w = a\#b\#c$), then $uv^0xy^0z \notin L$.
   - Next, as $|\#b\#| > p$, the vxy part cannot span parts from both a and c.
   - Hence, while pumping w, we cannot change the (arithmetic) value of a or c as we could only change one of these values.
   - As both a and c do not contain leading 0s, we cannot change either of them.
   - Moreover, note that we can neither add nor remove a 0 to/from b as c is fixed.
   - Finally, observe that the number of # signs in w is fixed.
5. In conclusion, there is no split $w = uvxyz$ that satisfies all criteria of the tandem-pumping lemma – a contradiction to p being a valid tandem-pumping length.
6. Consequently, L cannot be context-free. $\square$

Miscellaneous: If we chose any string $w = a\#b\#c$ with $1 \in b$, i.e. $b = b_1 1 b_2$, it would be tandem-pumpable.
Proof: Let $b = b_1 1 b_2$, then w is tandem-pumpable for the split $w = uvxyz$ where $u = a\#b_1$, $v = 1$, $x = \varepsilon$, $y = \varepsilon$, and $z = b_2\#c$.

4 Queueing Networks (22 points)

We have two tokens, which transit between queues a, b and c (each modeled as M/M/1) in this Continuous Time Queueing Network:



a) [6] Draw a Markov chain which precisely models the situation of the two tokens, without assuming Gordon-Newell's Theorem. (For example, if one token is in queue a, and one token is in queue b, we call this the state (1,1,0).)

b) [4] Does the Markov chain always have a unique stationary distribution, given that some rates might be 0? Explain why (not).

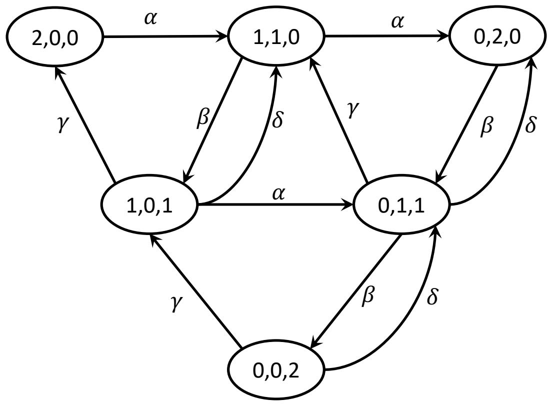
c) [4] What is the stationary distribution when $\alpha = \beta = \delta = 1$ and $\gamma = 0$?

d) [4] What is the stationary distribution when $\alpha = 1$, $\beta = 1$, $\gamma = 1$, and $\delta = 0$.

e) [4] If $\alpha = \beta = \gamma = \delta = 1$, in the steady state, what do you think is the least likely allocation of tokens (i.e., a state in your solution to (a))? Explain convincingly with a sentence, without computing probabilities.

Model solution

a) See the graph below:



Each state consists of three numbers: (#tokens_in_a, #tokens_in_b, #tokens_in_c)

b) The Markov chain has a unique stationary distribution except: (i) $\alpha = \beta = 0$. Stationary distributions are (2, 0, 0), (1, 1, 0), (0, 2, 0). (ii) $\alpha = \beta = \gamma = \delta = 0$. The Markov chain is stationary at its initial states.

c) When $\gamma = 0$, and $\alpha = \beta = \delta = 1$, the Markov chain in (a) reduces to three states: (0,2,0), (0,1,1), and (0,0,2). It is obvious to see that in this case $\pi_{(0,0,2)} = \pi_{(0,2,0)}$, and from state (0,1,1) we know $2\pi_{(0,1,1)} = \pi_{(0,0,2)} + \pi_{(0,2,0)}$. Hence, $\pi_{(0,1,1)} = \pi_{(0,0,2)} = \pi_{(0,2,0)} = \frac{1}{3}$.

d) We use $\pi_0, \pi_1, \pi_2, \pi_3, \pi_4$ and π_5 to denote the stationary distribution for state (2,0,0), (1,1,0), (0,2,0), (1,0,1), (0,1,1) and (0,0,2), respectively.

When $\alpha = 1$, $\beta = 1$, $\gamma = 1$, and $\delta = 0$, the nodes of Markov chain in (1) is symmetric. Hence, we have $\pi_0 = \pi_1 = \pi_2 = \pi_3 = \pi_4 = \pi_5 = \frac{1}{6}$.

e) The least likely allocations of tokens are (0, 0, 2), (2, 0, 0), (1, 0, 1) (Giving any of these three states is considered correct). By counting the input and output edges of the states, we can find these three states have the largest "delta" edges.

Alternative explanation for (2, 0, 0): From the original queueing networks we can observe that the tokens tends to stay more at state b or c, when $\alpha = \beta = \gamma = \delta = 1$. Hence, the two tokens stay at state a at the same time should be the least unlikely.

5 Card Game (18 points)

A dealer gives you cards, one card after another. Each card i has a value (an arbitrary positive integer), no two cards have the same value. You want to collect as many cards as possible. Only the number of cards matters, not the values on the cards. But there is a restriction: You can only collect a card with a higher value than the highest value you have collected (in your hand) already. These are the base rules of the card game. All the variants below follow these base rules. We study three variants of this game.

Variant 1:
You do not know how many cards the dealer has; at some point the dealer will say that there are no more cards. You also do not know what the values of the cards are.
You have two possible actions after the dealer deals you a new card i: 1) add card i to your collection in your hand; or 2) discard card i.

a) `[6]` Does there exist a strict constant-competitive online algorithm? Prove your claim.

Variant 2:
Before the dealer gives you any cards, the dealer shuffles the cards and randomizes the order of cards. You use a greedy algorithm to collect cards: you collect every card that the dealer gives you (if possible).

b) `[6]` What is the expected number of cards you can collect with the greedy algorithm if the dealer gives you n cards?

Variant 3:
You are now required to collect cards consecutively. In other words, if you have collected card i and skipped card $i+1$, you cannot collect any card j, with $j > i+1$. Moreover, your friend Bob now helps you. Your actions are: 1) drop all cards that you have collected and collect the new card 2) give the cards you collected to Bob and collect the new card. However, when you give Bob cards, Bob cannot keep the cards you gave to him previously, and Bob must drop your previous collection. In this variant, the cards are not shuffled, and you do not know how many cards the dealer has. You want to collect as many cards as possible (Bob and you have together).

c) [6] Does there exist a strict constant-competitive online algorithm for this scenario? Prove your claim.

Model solution

a) There is no such algorithm. We show this by designing an input sequence. Assume that card 1 has value v_1. Clearly, Alg must collect card 1 as otherwise Opt will collect card 1 and no other cards come, while the competitive ratio will be arbitrarily large.

After Alg accepts card 1 with value v_1, there will be a sequence of cards with values from 1 to $v_1 - 1$, which Alg will have to skip all of them. However, Opt will skip card 1 and collect all the others. Thus, we have $utility_{OPT} = v_1 - 1$ and $utility_{Alg} = 1$. For any given constant r, we can choose v_1 so that $utility_{OPT} > r\cdot$ The competitive ratio will be larger than r.

b) The expected number of cards that can be collected is $\sum_{i=1}^{n} \frac{1}{i}$.

We prove this by induction.

If $n = 1$, then the expected number of card that can be collected with the greedy algorithm is 1.

Then, assume that the expected number of cards that the greedy algorithm collects with $n - 1$ cards is $N(n-1) = \sum_{i=1}^{n-1} \frac{1}{i}$. Consider that we add a card with a value lower than other $n - 1$ cards to the sequence randomly. With probability $\frac{1}{n}$, the new card is inserted at the beginning of the sequence and the number of collected cards increases 1. Otherwise, the card is inserted in the middle of the sequence and cannot be collected by the greedy algorithm with probability $\frac{n-1}{n}$.

Therefore, the expected number of cards that the greedy algorithm collects with n cards is $N(n) = \frac{1}{n}(N(n-1) + 1) + \frac{n-1}{n}N(n-1) = N(n-1) + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i}$.

c) There is a greedy algorithm with a competitive ratio of 2.

You collect cards until it is not possible to collect new cards. Then you compare the number of cards that you have collected in this round with the number of cards that Bob has. If you have more cards than Bob, then you give all your cards to Bob. Otherwise, you drop all cards and start another round of collecting.

Lower bound: any sequence of cards.

Upper bound: you will collect all consecutive ascending sequences and compare the length with the register (Bob), and only the longest one will be kept in the register until the end of the game.

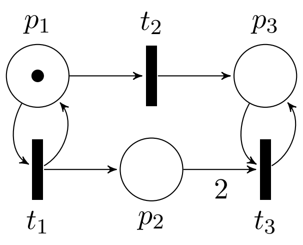
6 Multiple-Choice on Model Checking (6 points)

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

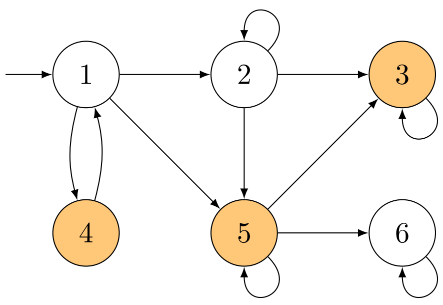
6.1 General Questions `[2 points]`

Which of the following statements are true?	**TRUE**	**FALSE**
a) Model-checking is a technique to verify that a system's model matches its implementation. *Model-checking verifies if a model satisfies a specification.*	☐	☑
b) Standard Petri nets are strictly more expressive than Finite Automata. *Petri nets can have an unbounded amount of tokens.*	☑	☐

6.2 Petri Nets `[2 points]`

Consider the following Petri net **P**: 	**TRUE**	**FALSE**
a) A valid execution trace of a Time Petri net is also a valid trace of the non-time version of the same Petri net. *A transition in an untimed Petri net can fire at any time. The Time Petri net puts constraints on the transitions.*	☑	☐
b) The transition t_1 in the given Petri net **P** is L$_3$-Live. *In the firing sequence $[t_1, t_1, t_1, \ldots]$, the transition t_1 fires infinitely often.*	☑	☐

6.3 Computation Tree Logic `[2 points]`

Consider the following automaton **A**, where the property p is satisfied only in states 3, 4 and 5:  In the following, $[\![\, \phi \,]\!]$ denotes the set of states which satisfy property ϕ. For example for automaton **A**, $[\![\, p \,]\!] = \{\, 3,\ 4,\ 5 \,\}$.	**TRUE**	**FALSE**
a) For automaton **A**, $[\![\, \mathbf{EX}\,((\mathbf{AX}\ p) \vee (\mathbf{AF}\ \overline{p}))\,]\!] = \{\, 2,\ 3,\ 5 \,\}$. *The correct answer is* $\{\, 1,\ 2,\ 3,\ 4,\ 5,\ 6 \,\}$	☐	☑
b) Automaton **A** satisfies **AF** p. *The sequence of states* $[\, 1,\ 2,\ 2,\ \ldots \,]$ *never reaches a state that satisfies* p.	☐	☑

7 Binary Decision Diagram (10 points)

a) [5] Given the Boolean expression of function f and the ordering of variables $x_1 \prec x_2 \prec x_3 \prec x_4$, 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.

$$f(x_1, x_2, x_3, x_4) = \Big(\; (\; \overline{x_1} + x_2 \;) \cdot x_3 + \overline{x_4} \; \Big) \cdot \Big(\; x_1 \cdot \overline{x_2} + \overline{x_3} + x_4 \; \Big)$$

b) `[2]` Consider the BDD of the function g in Figure 1. Express g as a boolean function.

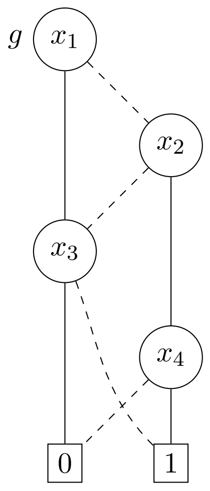


Figure 1: BDD of the Boolean function g

c) [3] Consider the ROBDD of the function h in Figure 2. Construct the ROBDD of the function $h'(x_1, x_2, x_3) = \exists(x_4, x_5) : h(x_1, x_2, x_3, x_4, x_5)$. Merge all equivalent nodes, including the leaves.

Note: Use solid lines for *True* arcs and dashed lines for *False* arcs.

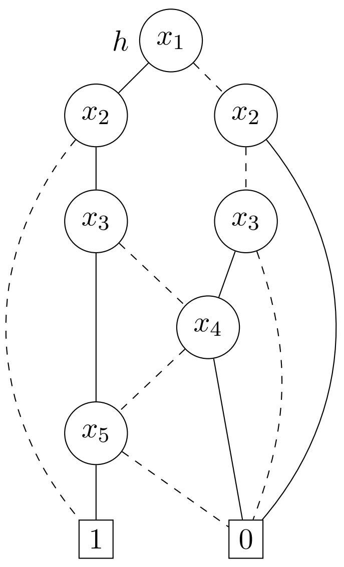
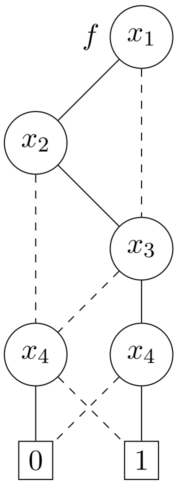


Figure 2: ROBDD of the Boolean function h

Model solution

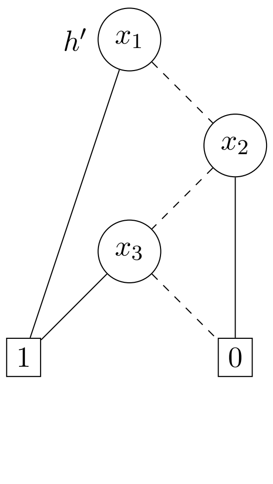
a) The BDD of f is shown below;



b) The function g can be expressed by the following boolean expression

$$g(x_1, x_2, x_3, x_4) = x_1 \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot x_4$$

c) Finally, the ROBDD of h' is shown below;



8 Computation Tree Logic (8 points)

Throughout this question, we will consider the execution of a program. The following properties are defined on the state of the program:

- r: The program is in an initial state.
- e: The program is in an error state.
- i: The program is reading an input from the user.
- t: The program is in a termination state.

In this question, a CTL formula can contain the CTL operators, as well as logical operators like $\wedge$ (and), $\vee$ (or), $\neg$ (not), and $\implies$ (implication), and the properties r, e, i, and t.
Note: The two sub-questions (8.1 and 8.2) are independent.

8.1 From Text to CTL `[4 points]`

For each of the following sentences, write a CTL formula that describes the sentence.

a) `[1]` The set of states from which, in every future, there always exists a possibility to reach the initial state.

b) `[3]` The set of states where an input is read, and there exists a successor state (i.e., a specific input from the user), for which every future execution will result in the program terminating without reaching an error state.

8.2 From CTL to Text [4 points]

For each of the following CTL formulas, describe the set of states for which the CTL expression is satisfied.

a) [2] $t \implies \mathbf{AG}\, t$

b) [2] $r \wedge \big(\neg e\ \mathbf{AU}\ i\big)$

Model solution

From Text to CTL

a) $\mathbf{AG}\ \mathbf{EF}\ r$

b) Multiple valid solutions [3]:

- $i \wedge \mathbf{EX}\big(\mathbf{AF}\ t \wedge \mathbf{AG}\ \neg e\big)$
- $i \wedge \mathbf{EX}\big(\mathbf{AF}\ t \wedge \neg e\ \mathbf{AU}\ t\big)$

Solutions that only give a single point [1]:

- $i \wedge \mathbf{EX}\big(\neg e\ \mathbf{AU}\ t\big)$
- $\mathbf{EX}\big(\mathbf{AF}\ t \wedge \mathbf{AG}\ \neg e\big)$
- $\mathbf{EX}\big(\mathbf{AF}\ t \wedge \neg e\ \mathbf{AU}\ t\big)$

From CTL to Text

a) $t \implies \mathbf{AG}\ t$ describes all states in which the program has not terminated ($\neg t$), **plus** those in which the program has terminated and remains terminated in every possible future.

b) $r \wedge \big(\neg e\ \mathbf{AU}\ i\big)$ describes all initial states for which there exists no future where the program can reach an error state **before** it is reading an input form the user. This means it includes the initial state if there exists no future where an error state is reached.

9 Petri Nets (16 points)

Throughout this question, we use the following notations.

- $M^\top = [m_1, m_2, m_3, \ldots]$ and $U^\top = [u_1, u_2, u_3, \ldots]$ are marking and firing vectors of **P**, respectively.
- m_i denotes the number of tokens in place p_i.
- u_i denotes the number of firings of transition t_i.

Note: The two sub-questions (9.1 and 9.2) are independent.

9.1 Reachability `[8 points]`

Let us consider the Petri net **P1** in Figure 3.

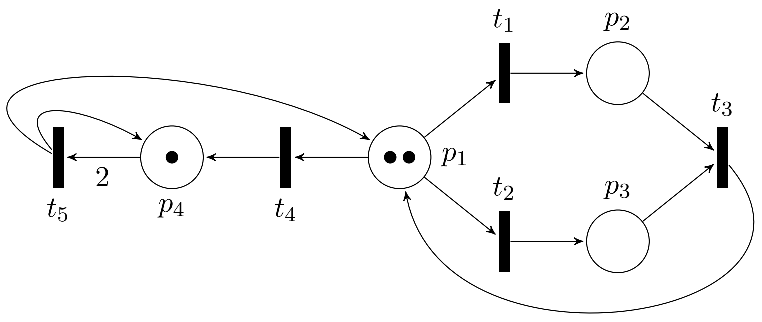


Figure 3: Petri net **P1** – Circles, dots and bars represent places, tokens and transitions, respectively. Edge weights are marked close to the edge when they are different from 1.

a) `[2]` Derive the incidence matrix A of the Petri net **P1** from Figure 3.

b) [2] Given initial marking $M_0^\top = [2, 0, 0, 1]$, is the Petri net deadlock-free? If so, provide a brief proof. If not, provide a valid firing sequence that leads the net into a deadlock.

c) [2] Consider the firing vector $U_S^\top = [1, 1, 0, 1, 1]$, where S denotes a firing sequence containing the firing of t_1, t_2, t_4 and t_5 once.
Use the incidence matrix and the state equation of the Petri net **P1** to compute the marking $M_1^\top$ obtained from the initial marking $M_0^\top = [2, 0, 0, 1]$ after firing S.

d) [2] Given initial marking $M_0^\top = [2, 0, 0, 1]$ and assume $M_1^\top$ computed in (c) is a valid marking (i.e., all elements are non-negative). Is it true that all firing sequences S with firing vector $U_S^\top = [1, 1, 0, 1, 1]$ are feasible? Justify your answer.

Model solution

a) $A = \begin{bmatrix} -1 & -1 & 1 & -1 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$

b) Deadlocks with $[0, 1, 0, 1]$ or $[0, 0, 1, 1]$. Valid solutions are:

- t_1, t_2, $t_3\, t_1$
- t_1, t_2, $t_3\, t_2$
- t_2, t_1, $t_3\, t_1$
- t_2, t_1, $t_3\, t_2$
- Any of the above, that have $(t_4,\, t_5)^*$ either at the beginning, or right after t_3.

c)

$$M_1 = A \cdot T_S + \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

d) Not necessary a valid firing sequence. If t_1 and t_2 fire first, then t_4 can never fire.

9.2 Coverability `[8 points]`

Let us consider the Petri net **P2** in Figure 4.

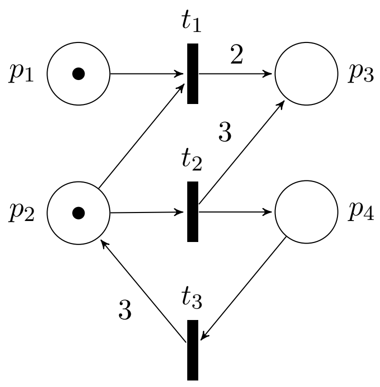
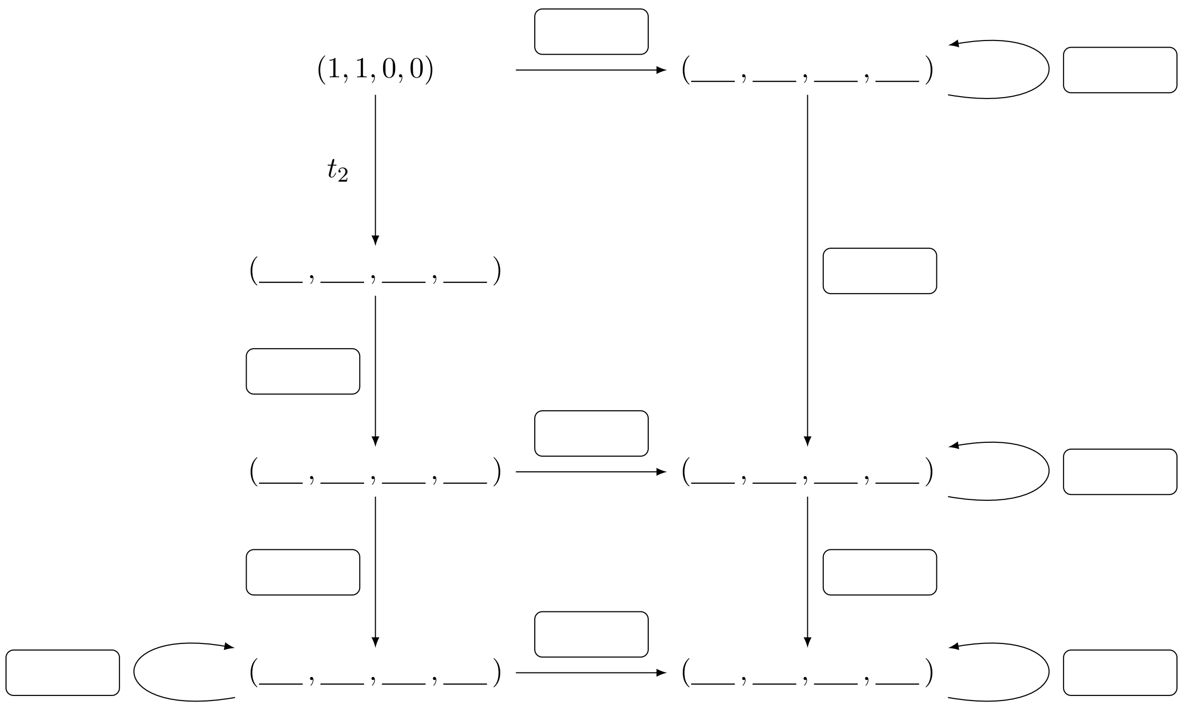


Figure 4: Petri net **P2** – Circles, dots and bars represent places, tokens and transitions, respectively. Edge weights are marked close to the edge when they are different from 1.

a) `[4]` Construct the coverability graph of the Petri net **P2** by filling out the provided skeleton.

Notes:

- Some transitions in the skeleton are not needed. Clearly mark the transitions you do not use by crossing out the transition arrow and the box.
- The coverability graph is obtained from the coverability tree by merging nodes with the same marking.



b) [1] Is $M^\top = [2, 13, 27, 2]$ reachable? Justify.

c) [3] Is $M^\top = [0, 18, 26, 0]$ reachable? Justify.

Model solution

a)

$$\begin{tikzcd}
(1,1,0,0) \arrow[r, "t_1"] \arrow[d, "t_2"] & (0,0,2,0) \\
(1,0,3,1) \arrow[d, "t_3"] & \\
(1,\omega,\omega,0) \arrow[r, "t_1"] \arrow[d, "t_2"] & (0,\omega,\omega,0) \arrow[d, "t_2"] \\
(1,\omega,\omega,\omega) \arrow[r, "t_1"] \arrow[loop left, "{t_2,\ t_3}"] & (0,\omega,\omega,\omega) \arrow[loop right, "{t_2,\ t_3}"]
\end{tikzcd}$$

b) No, $M^\top = [2, 13, 27, 2]$ is not reachable from $M_0^T = [1, 1, 0, 0]$, because it is not covered in the coverability graph. A different, valid explanation is that place p_1 has only outgoing arcs to transitions, and no incoming arcs from any transitions. So any transition firing either removes tokens from p_1, or does not change the token count in p_1.

c) No, $M^\top = [0, 18, 26, 0]$ is not reachable from $M_0^T = [1, 1, 0, 0]$. If p_1 still has one token, then p_3 contains $3k$ tokens, where k is the number of times t_2 has fired. Since p_4 has no tokens (still assuming p_1 has a single token), place p_2 contains $1 + 2k$ tokens. If at some point t_1 fires, then p_2 contains one token less, and p_3 contains two tokens more. Hence, at the end, if p_1 and p_4 both have no tokens, then $[0, m_2, m_3, 0]^T$ is a valid marking only if there exists a $k \in \mathbb{N}_0$, such that $m_2 = 2k$ and $m_3 = 3k + 2$. This does not hold for $M^T = [0, 18, 26, 0]$.