Chair of Efficient Algorithms
School of Computation, Information and Technology
Technical University of Munich

Esolution
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Information Theory and Theory of Computation

Exam:	INHN0013 / Bonus Test 1	**Date:**	Wednesday 29th October, 2025
Examiner:	Prof. Kobourov	**Time:**	09:00 – 10:00

Sample Solution

Working instructions

- This exam consists of **14 pages** with a total of **7 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 65 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - None
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.

- *Mark correct answers with a cross* ☒
- *To undo a cross, completely fill out the answer option* ■
  To re-mark an option, use a human-readable marking ✕■

Left room from _____ to _____ / Early submission at _____

Problem 1

(10 credits)

0 1 2 3 4 5 6 7 8 9 10

Version A

Formally prove by construction that regular languages are closed under the complement operation. That is, given a DFA for some regular language, modify it so that it recognizes precisely the complement of the language, and argue the claim using the formal definition of computation for a DFA.

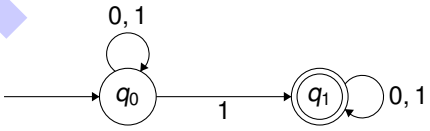
Let $\mathcal{L}$ be arbitrary regular language. According to definition of regular language, there exists a DFA $\mathcal{M}$, defined by $(Q, \Sigma, \delta, q_0, F)$ that recognizes it. Then the desired DFA $\mathcal{M}'$ that recognizes the complement of $\mathcal{L}$ can be defined as $(Q, \Sigma, \delta, q_0, Q \setminus F)$.
$\mathcal{M}'$ recognizes complement of $\mathcal{L}$ because: for arbitrary string s made of given alphabet, any DFA with Q, Σ, δ, q_0 would end up in the same state q_i for this input(dictated by formal definition of computation of DFAs). Thus :

- In the case that s is in $\mathcal{L}$, s would be accepted by $\mathcal{M}$, thus q_i is in F, thus rejected by $\mathcal{M}'$.
- In the case that s is not in $\mathcal{L}$, s would be rejected by $\mathcal{M}$, thus q_i is not in F, but still in Q, thus accepted by $\mathcal{M}'$.

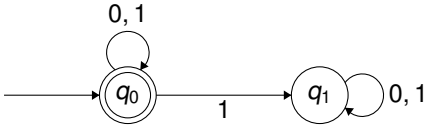
In conclusion, for arbitrary regular language, a DFA that recognizes its complement can be constructed, thus its complement must also be regular.

Consider a proof by construction that regular languages are closed under the complement operation, but this time based on modifying an NFA for the language, using the same method as that for DFAs above. Show that this approach does not lead to a correct proof.

Consider following NFA that recognizes the language $\{w | w \text{ contains at least one } 1\}$ with alphabet of $\{0, 1\}$.



By flipping accept and non-accept states, we have the following NFA that accepts all legal strings.



Version B

Formally prove by construction that regular languages are closed under the intersection operation. That is, given DFAs for two regular languages, create a new DFA that recognizes precisely the intersection of the two languages, and argue the claim using the formal definition of computation for a DFA.

Sample Solution

Let L_1 and L_2 be two arbitrary regular languages, and let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs recognizing L_1 and L_2, respectively. Idea: Run M_1 and M_2 in parallel on the same input and accept if both M_1 and M_2 accept. Consider $M = (Q, \Sigma, \delta, q_0, F)$ defined as follows:

- $Q = Q1 \times Q2$
- $q0 = (q1, q2)$
- $\delta((p1, p2), a) = (\delta_1(p1, a), \delta_2(p2, a))$
- $F = F_1 \times F_2$

M recognizes $L_1 \cap L_2$ because for the following reason. Let s be an arbitrary string made of given alphabet, and let f_1, f_2 be the final states that M_1 and M_2 will end up in for input s, respectively. Then, by construction M ends up in (f_1, f_2) for input s.

- If s is in $L_1 \cap L_2$, then by definition, both M_1 and M_2 accept s, which implies $f_1 \in F_1$ and $f_2 \in F_2$. Hence, $(f_1, f_2) \in F_1 \times F_2$, i.e. s is accepted by M.
- If s is not in $L_1 \cap L_2$, then by definition, either M_1 or M_2 (or both) don't accept s, which implies $f_1 \notin F_1$ or $f_2 \notin F_2$. Hence, $(f_1, f_2) \notin F_1 \times F_2$, i.e. s is not accepted by M.

In conclusion, for arbitrary two regular languages, a DFA that recognizes their intersection can be constructed, thus their intersection must also be regular.

Sample Solution

Problem 2

(10 credits)

Build FAs for the following languages

0 1 2 3 4 5

a) $L_1 = \{xy | x \in A \text{ and } y \notin A\}$ where A is some regular language

- Build DFA $\mathcal{M}_1$ that recognizes $x|x \in A$, DFA $\mathcal{M}_2$ that recognizes $y|y \notin A$ can be constructed as states in question 1 part (a).
- Combine $\mathcal{M}_1$ and $\mathcal{M}_2$ by adding an ϵ transition from all accept states of $\mathcal{M}_1$ to initial state of $\mathcal{M}_2$.
- Make initial state of $\mathcal{M}_1$ as initial state of combined FA, and make only final states of $\mathcal{M}_2$ as final states of combined FA.

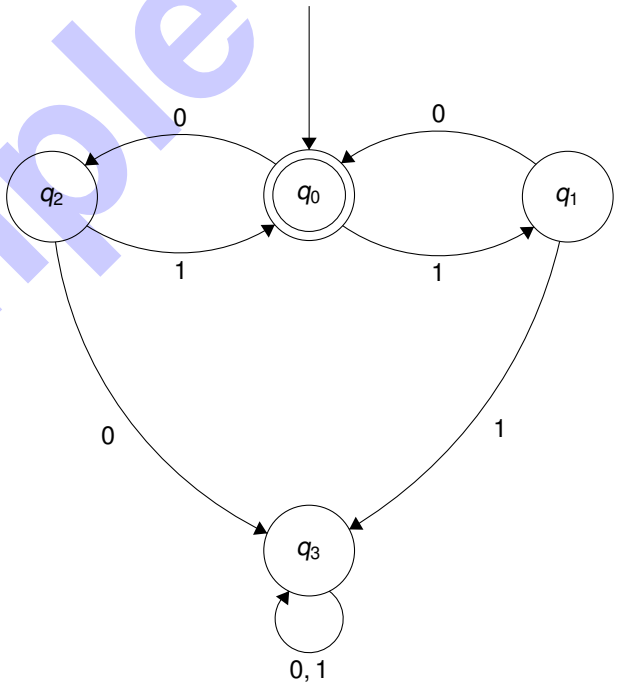
0 1 2 3 4 5

b)

Version A

$L_2 = \{w | w \in \{0 \cup 1\}^*$ with equal number of 0's and 1's and every prefix of w has at most one more of either 0's or 1's $\}$. Recall that we say x is a prefix of y if there exists z such that $xz = y$.

q_0 : #0 = #1 so far
q_1 : #1 = 1 + #0 so far
q_2 : #0 = 1 + #1 so far
q_3 : has a prefix with #0 $\geq$ 2 + #1 *OR* #1 $\geq$ 2 + #0

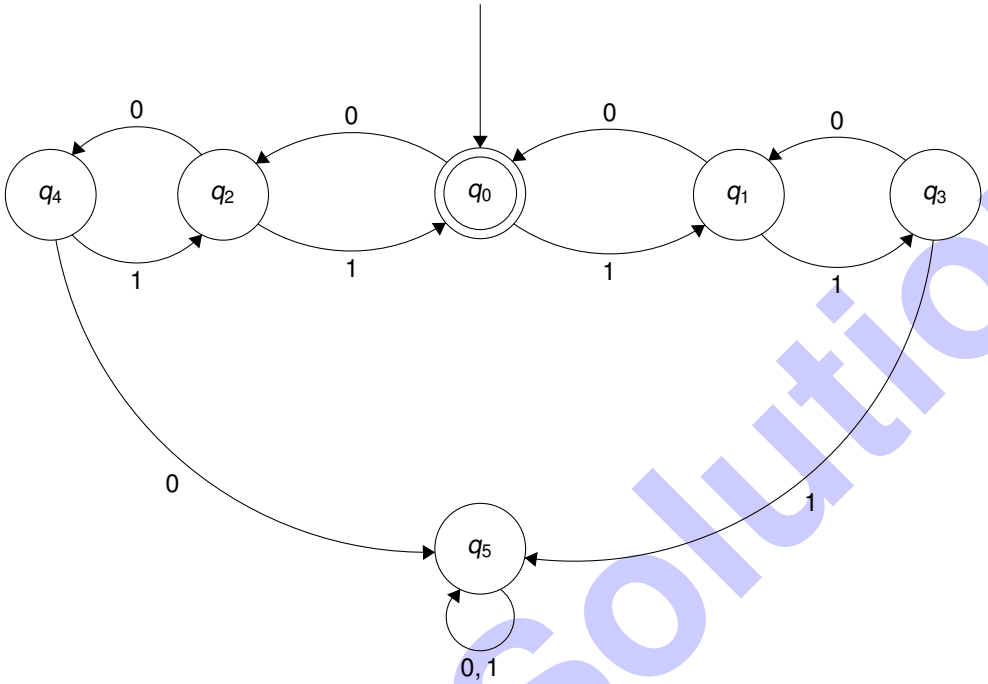


Version B

$L_2 = \{w | w \in \{0 \cup 1\}^*$ with equal number of 0's and 1's and every prefix of w has at most two more of either 0's or 1's $\}$. Recall that we say x is a prefix of y if there exists z such that $xz = y$.

Sample Solution

q_0 : #0 = #1 so far
q_1 : #1 = 1 + #0 so far
q_2 : #0 = 1 + #1 so far
q_3 : #1 = 2 + #0 so far
q_4 : #0 = 2 + #1 so far
q_5 : has a prefix with #0 $\geq$ 3 + #1 *OR* #1 $\geq$ 3 + #0



Sample Solution

Problem 3

(10 credits)

0 1 2 3 4 5

a) Produce regular expression for $L_1 = \{w | w$ starts with a 1 and has at most one $0\}$.

$R_1 = 1((1^*01^*) \cup (1^*))$

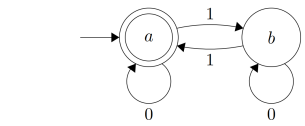
b)

Version A

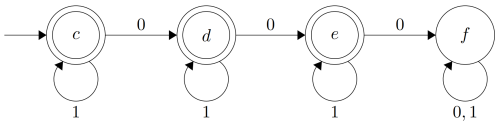
0 1 2 3 4 5

The language $L_2 = \{w | w$ has an even number of 1s or at most two 0s$\}$ is the union of two simpler languages. Construct DFAs for the simpler languages and combine them to obtain a state diagram of a DFA for the language. Here $\Sigma = \{0, 1\}$. (Hint: This means you should have 3 DFAs for your answer.)

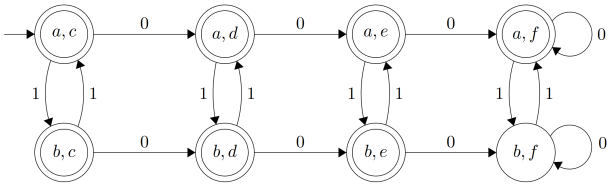
$D_1 =$



$D_2 =$



$D_1 \cup D_2 = L_2$

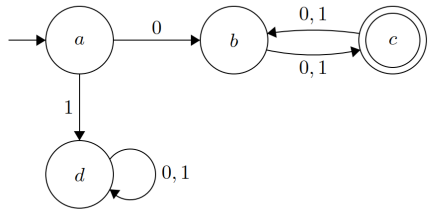


Version B

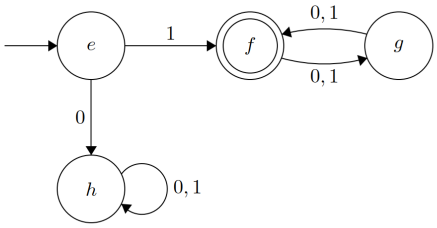
The language $L_2 = \{w | w$ starts with 0 and has even length or starts with 1 and has odd length$\}$ is the union of two simpler languages. Construct DFAs for the simpler languages and combine them to obtain a state diagram of a DFA for the language. Here $\Sigma = \{0, 1\}$. (Hint: This means you should have 3 DFAs for your answer.)

Sample Solution

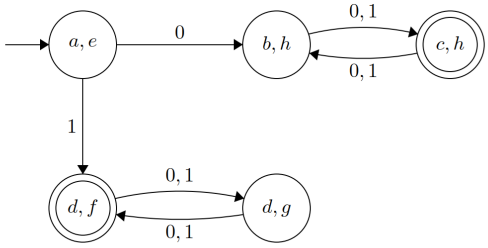
$D_1 =$



$D_2 =$



$D_1 \cup D_2 = L_3$

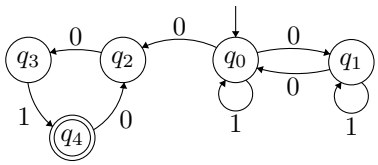


Sample Solution

Problem 4
(5 credits)

0 1 2 3 4 5

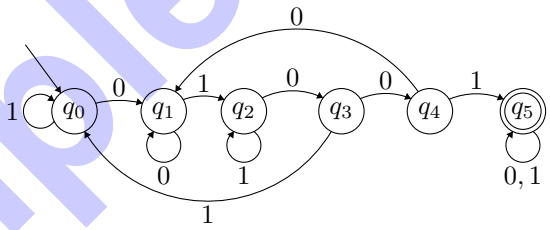
Version A
For the language $L = \{uv \mid u$ has an even number of 0's and is not empty, and $v \in (001)^+\}$, an NFA has been constructed, supposedly recognizing the language. Verify whether this is correct and if not identify the bugs, explain what the problems are, and show how to correct them. You may add, delete, or modify states and transitions, but do not construct an entirely new machine. The alphabet used here is $\{0, 1\}$.



This machine is wrong, it wrongfully accepts string 001. Fix:

- Add the new state q_5
- Make q_5 the new start state
- Add $\delta(q_5, 0) = q_1$
- Add $\delta(q_5, 1) = q_0$

Version B
For the language $L = \{w \mid w$ has 01001 as a substring$\}$, an NFA has been constructed, supposedly recognizing the language. Verify whether this is correct and if not identify the bugs, explain what the problems are, and show how to correct them. You may add, delete, or modify states and transitions, but do not construct an entirely new machine. The alphabet used here is $\{0, 1\}$.



The NFA is incorrect. It wrongfully accepts string 011001 and misses string 0101001. It can be fixed by:

- Remove $\delta(q_2, 1) = q_2$
- Add $\delta(q_2, 1) = q_0$
- Remove $\delta(q_3, 1) = q_0$
- Add $\delta(q_3, 1) = q_2$

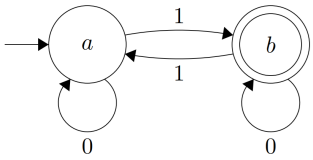
0 1 2 3 4

Problem 5
(12 credits)

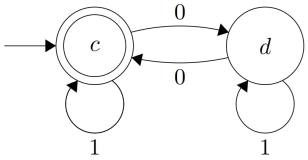
a) The language $L = \{w | w$ has an odd number of 1s and an even number of 0s$\}$ is the intersection of two simpler languages. Construct DFAs for the simpler languages and combine them to obtain a state diagram of a DFA for the language. Here $\Sigma = \{0, 1\}$. (Hint: This means you should have 3 DFAs for your answer.)

Sample Solution

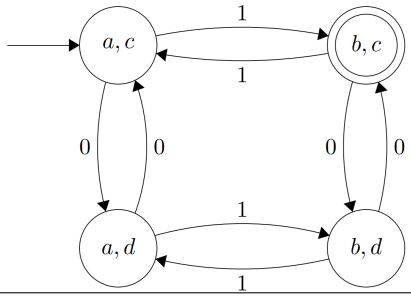
$D_1 =$



$D_2 =$



$D_1 \cap D_2 = L_2$

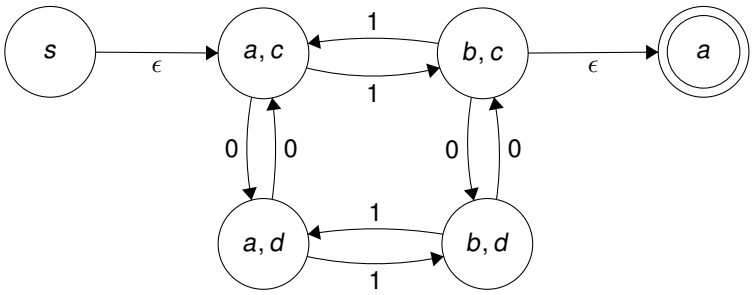


Sample Solution

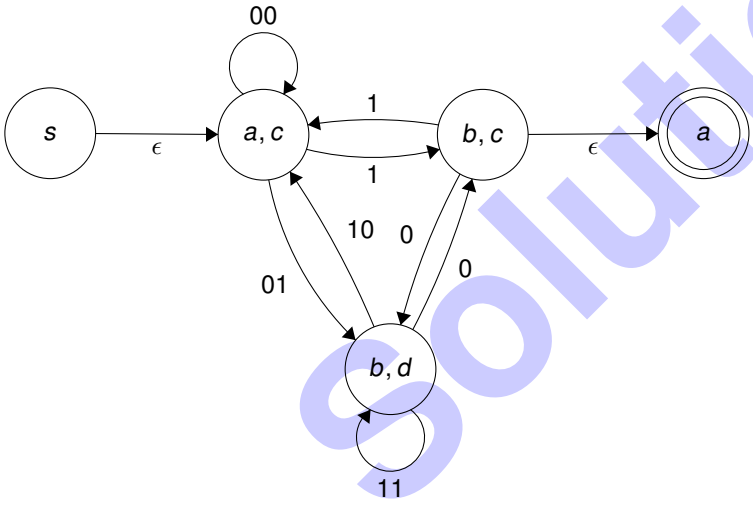
0 1 2 3 4 5 6 7 8

b) Convert your answer in (a) to regular expression using the GNFA-based approach. Show intermediate steps, not just the answer.

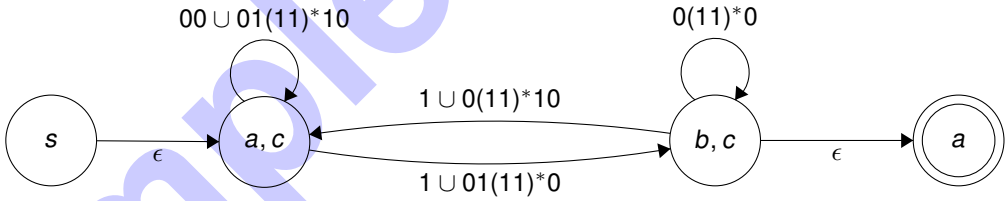
1. Add new start and accept states



2. remove state (a,d)

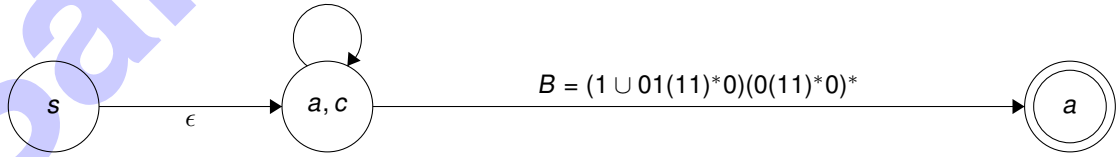


3. remove state (b,d)

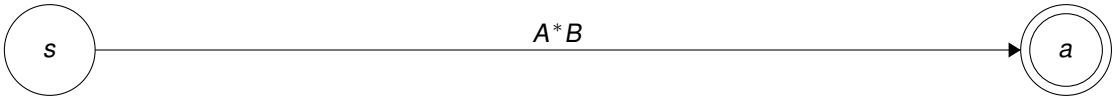


4. remove state (b,c)

$A = 00 \cup 01(11)^*10 \cup (1 \cup 01(11)^*0)(0(11)^*0)^*(1 \cup 0(11)^*10)$



5. remove state (a,c)



Sample Solution

Problem 6

(8 credits)

0 1 2 3 4 5 6 7 8

Version A

Prove or disprove that the language $L = \{w | w$ has the same number of 0's and 1's$\}$ is regular.

Not regular. Proof using pumping lemma.

- Assume L_1 is regular.
- Since pumping lemma is a necessary condition for a language to be regular, we know pumping lemma holds for L_1.
- According to pumping lemma, we can find integer p, such that arbitrary string w in L_1 with length of at least p has following property:
  1. w can be divided into three parts, and $xy^iz \in L_1$ for all $i \geq 0$
  2. $|y| > 0$
  3. $|xy| \leq p$
- Consider string 0^p1^p. For every possible partitioning, y must only contain 0s. So for $i > 1$, there will be more 0 than 1 in pumped string.
- Thus pumping lemma does not hold for L_1. Contradicts with the second step.
- Thus assumption is false, L_1 is not regular.

Sample Solution

Version B

Prove or disprove that the language $L = \{w | w$ is not a palindrome$\}$ is regular.

Not regular. Proof using closure properties and pumping lemma.

- Assume $L = \{w \mid w \text{ is not a palindrome}\}$ is regular.
- A known property of regular languages is that they are closed under complement.
- This means that if L is regular, its complement L^c must also be regular.
- The complement of L is $L^c = \{w \mid w \text{ is a palindrome}\}$.
- We can now prove that L^c is not regular using the pumping lemma.
- Assume L^c is regular. Let p be the pumping length.
- Consider the string $w = 0^p10^p$. This string is in L^c (it's a palindrome) and its length is $2p + 1 \geq p$.
- According to pumping lemma, we can find integer p, such that arbitrary string w in L^c with length of at least p has following property:
  1. w can be divided into three parts, and $xy^iz \in L^c$ for all $i \geq 0$
  2. $|y| > 0$
  3. $|xy| \leq p$
- Because $|xy| \leq p$, x and y must be part of the initial 0^p block. Thus, y consists of one or more 0s (i.e., $y = 0^k$ where $k \geq 1$).
- Now, let's pump the string with $i = 2$. The new string is $xy^2z = 0^{p+k}10^p$.
- Since $k \geq 1$, the number of 0s before the 1 is different from the number of 0s after the 1. This new string is not a palindrome.
- Therefore, $xy^2z \notin L^c$. This is a contradiction, as the pumping lemma guarantees the pumped string will be in the language.
- This contradicts our assumption that L^c is regular. So, L^c is not regular.
- Since L^c is not regular, our original assumption must be false, and L is not regular.

Problem 7 **Multiple Choice (10 credits)**

a) Every subset of a regular language is a regular language.

- ☐ True
- ☒ False

b) If $L_1 \subseteq L_2$ and L_1 is regular, then L_2 is also regular.

- ☐ True
- ☒ False

c) If $L_1 \subseteq L_2$ and L_2 is non-regular, then L_1 is also non-regular.

- ☐ True
- ☒ False

d) $(\epsilon \cup r)^* = r^*$

- ☒ True
- ☐ False

e) $(r \cup s)^* = r^* \cup s^*$

- ☐ True
- ☒ False

f) $r \cup (st) = (r \cup s)(r \cup t)$

- ☐ True
- ☒ False

g) $R = R\epsilon$, where R is any regular expression.

- ☒ True
- ☐ False

Sample Solution

h) $R = R\emptyset$, where R is any regular expression.

- [ ] True
- [x] False

i) $R \cup \epsilon = R$, where R is any regular expression.

- [ ] True
- [x] False

j) We can use the pumping lemma to prove that a language is regular.

- [ ] True
- [x] False

Sample Solution

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.

Sample Solution