CS 374 HW 2 Problem 2

quddus2, Aldo Sanjoto, Hieu Huynh

TOTAL POINTS

70 / 100

QUESTION 1

1 2A **0 / 20**

- **0 pts** correctly identifies the language as the set of all strings that differ from some string of the SAME LENGTH in L by AT MOST one bit (containment of L is implied)
- **3 pts** almost correct, did not indicate that for each x, the string in L it compares to has to have the same length
- **5 pts** indicates the language is the set of strings with hamming distance 1 from some string in L but does not expand on definition of hamming distance
- **20 pts** wrongly thinks L is the set of any two strings of hamming distance 1 from each other, or the set of all strings x of hamming distance 1 from a specific string in L

✓ - **20 pts wrong description of L**

- **20 pts** Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- **20 pts** unreadable
- **15 pts** IDK

What exactly is L_<=1? You said x is in Sigma*, but what does x has to do with L_<=1? Please be clearer in the future.

QUESTION 2

2 2B **20 / 20**

✓ - **0 pts correctly describes the language as the set of strings containing the substring 0001**

- **0 pts** correctly gives a regular expression (0+1)*0001(0+1)* or (1 + 01 + 001)*0000*1(0 + 1)* or other variations
- **20 pts** wrongly identifies the language
- **20 pts** Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- **20 pts** unreadable
- **15 pts** IDK
- **5 pts** confusing string with set of strings
- **2 pts** invalid symbol

QUESTION 3

3 2C **10 / 20**

- **0 pts** Correct construction of NFA, clear description of the meaning of two copies of M, clear description of new transitions
- **5 pts** explanation of the NFA construction is vague

✓ - **10 pts no explanation of NFA at all**

- **3 pts** minor mistakes (like missing one transition) in NFA
- **5 pts** Correct NFA but built from a simplified DFA, not following instruction
- **20 pts** Incorrectly NFA and construction idea
- **20 pts** Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- **20 pts** Unreadable
- **15 pts** IDK
- **15 pts** Missing NFA

You need to give explanation for you NFA.

QUESTION 4

4 2D **20 / 20**

✓ - **0 pts Correct formal definition of NFA and clear proof of correctness**

- **5 pts** proof of correctness is incomplete, such as showed NFA accepts L<=1 but doesn't show it doesn't accept other strings

- **10 pts** no proof of correctness
- **3 pts** No formal definition of NFA and description of how to construct NFA is mostly clear but lacks details such that what's the new start state of NFA
- **8 pts** No formal definition of NFA, description of how to construct NFA is vague
- **20 pts** Incorrect NFA construction and proof idea
- **20 pts** Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- **20 pts** Unreadable
- **15 pts** IDK

QUESTION 5

5 2E 20 / 20

✓ **- 0 pts Correct formal definition of NFA and clear proof of correctness**

- **5 pts** proof of correctness of NFA mingled with description of NFA and/or is incomplete, such as showed NFA accepts L<=k but doesn't show it doesn't accept other strings
- **10 pts** no proof of correctness of NFA
- **10 pts** Incorrect NFA construction
- **0 pts** proof by induction correct and complete
- **3 pts** proof of induction have minor mistakes
- **8 pts** proof of induction vague or missing steps, e.g., inductive step uses (D) to extend without specific details
- **20 pts** Incorrect NFA construction and proof idea
- **15 pts** IDK
- **20 pts** Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- **20 pts** unreadable
- **5 pts** NFA construction idea unclear

1 2A 0 / 20

- **0 pts** correctly identifies the language as the set of all strings that differ from some string of the SAME LENGTH in L by AT MOST one bit (containment of L is implied)
- **3 pts** almost correct, did not indicate that for each x, the string in L it compares to has to have the same length
- **5 pts** indicates the language is the set of strings with hamming distance 1 from some string in L but does not expand on definition of hamming distance
- **20 pts** wrongly thinks L is the set of any two strings of hamming distance 1 from each other, or the set of all strings x of hamming distance 1 from a specific string in L

✓ - **20 pts wrong description of L**

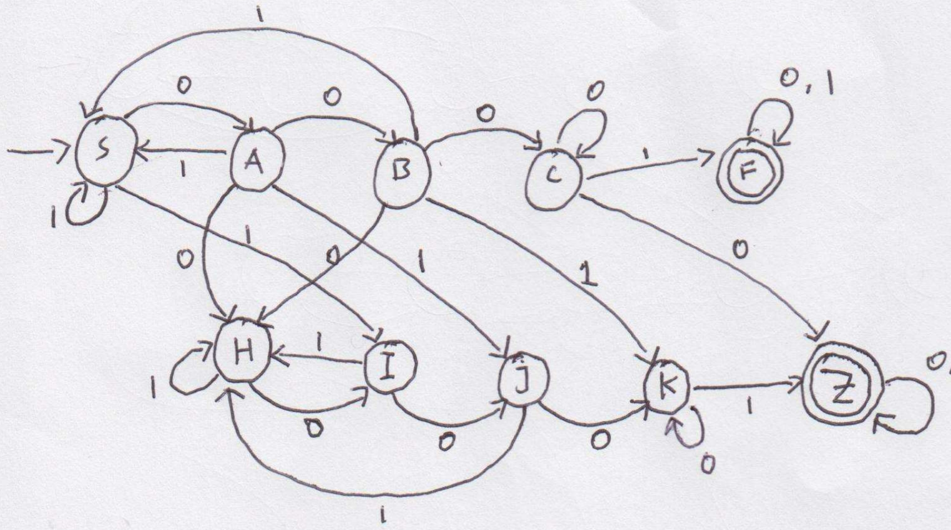
- **20 pts** Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- **20 pts** unreadable
- **15 pts** IDK

What exactly is L_<=1? You said x is in Sigma*, but what does x has to do with L_<=1? Please be clearer in the future.

2 a) $L_{\leq 1}$ is a language where for all strings X in Σ^* (where $\Sigma = \{0, 1\}$), there's at least one string y in the language L (where L is a subset of Σ^*) such that the hamming distance between x & y is less than or equal to 1.

b) The language L is every string that has the substring of 0001.

c)



2 2B 20 / 20

✓ **- 0 pts** **correctly describes the language as the set of strings containing the substring 0001**

- 0 pts correctly gives a regular expression (0+1)*0001(0+1)* or (1 + 01 + 001)*0000*1(0 + 1)* or other variations

- 20 pts wrongly identifies the language

- 20 pts Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.

- 20 pts unreadable

- 15 pts IDK

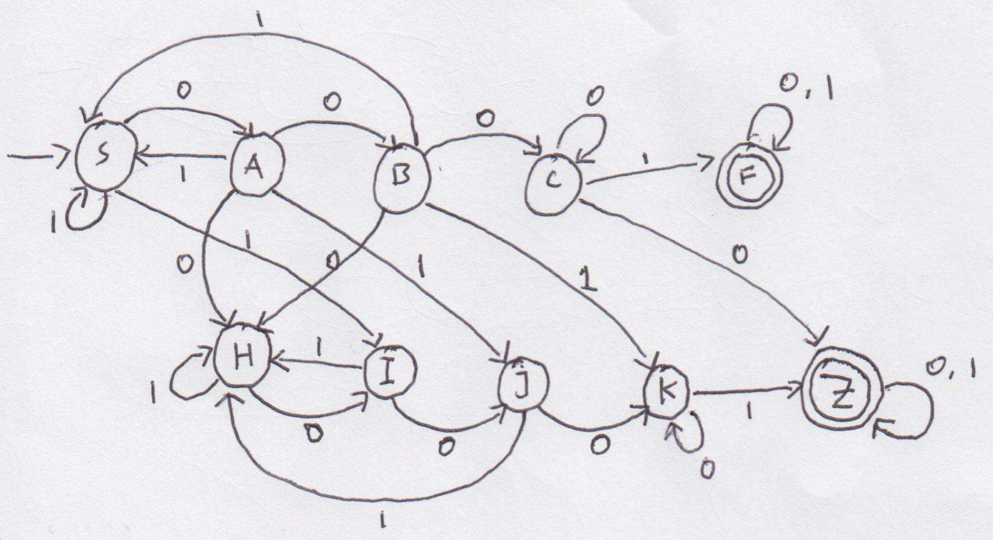
- 5 pts confusing string with set of strings

- 2 pts invalid symbol

2 a) $L_{\leq 1}$ is a language where for all strings X in Σ^* (where $\Sigma = \{0, 1\}$), there's at least one string y in the language L (where L is a subset of Σ^*) such that the hamming distance between x & y is less than or equal to 1.

b) The language L is every string that has the substring of 0001.

c)



3 2C 10 / 20

- 0 pts Correct construction of NFA, clear description of the meaning of two copies of M, clear description of new transitions
- 5 pts explanation of the NFA construction is vague
- ✓ - 10 pts **no explanation of NFA at all**
- 3 pts minor mistakes (like missing one transition) in NFA
- 5 pts Correct NFA but built from a simplified DFA, not following instruction
- 20 pts Incorrectly NFA and construction idea
- 20 pts Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- 20 pts Unreadable
- 15 pts IDK
- 15 pts Missing NFA

You need to give explanation for you NFA.

2D)

For the following explanation assume that X is a string that is part of the language L and y is a string which has the same length as X and compared to X only differs by one character. We can construct the NFA that accepts $L_{\leq 1}$ by creating two DFA's of L, called DFA 0 & DFA 1, where DFA0 has the starting state q_0 which also be the starting state of the NFA.

We observe all the transitions occuring in the original DFA. If there occurs some transition between state q_n & q_k (where $n \neq k$) due to some characters a then we draw a transition arrow from DFA0 state q_n to q_k of DFA 1 which can occur due to all characters in Σ except that some character a. (these transition that connect DFA0 to DFA1 is what leads to one NFA because start state is in DFA 0).

What this does is allows all NFA to continue on to the remaining same DFA states as the original DFA while accepting the single variation. So if the input to the NFA is y then after our variation we are forced to go to remaining states of the DFA as if the original DFA accepted the character variation. And from this point on, the only way we can reach the accepting state is for remaining characters in y to match X because there is no way for y to return to DFA0 which allows the variation because of the way we defined how transitions occur between DFA0 & DFA1.

4 2D 20 / 20

✓ - 0 pts Correct formal definition of NFA and clear proof of correctness

- 5 pts proof of correctness is incomplete, such as showed NFA accepts L<=1 but doesn't show it doesn't accept other strings

- 10 pts no proof of correctness

- 3 pts No formal definition of NFA and description of how to construct NFA is mostly clear but lacks details such that what's the new start state of NFA

- 8 pts No formal definition of NFA, description of how to construct NFA is vague

- 20 pts Incorrect NFA construction and proof idea

- 20 pts Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.

- 20 pts Unreadable

- 15 pts IDK

2e.) Let $M = (\Sigma, Q, s, A, \delta)$ be the DFA that accepts Some Language L. The following is the definition of ~~the~~ an NFA that accepts $L_{\leq k}$ for Some constant k.

$$Q_N = Q_m \times \{ \underbrace{n_0, n_1 \ldots n_k}_{\text{number of DFA copies}} \}$$ // So if we have ~~q~~ n states in the original DFA then
n_0 would have n states
n_1 would have n states
⋮
n_k would have n states

$S = (S_m, n_0)$ // this is q_0 in n_0

$A = \{ (q, n_i) \mid q \in A_m$ and $0 \leq i \leq k \}$

$\Sigma = \{0, 1\}$

For $0 \leq i \leq k-1$:

$$\delta_N((q, n_i), a) = \begin{cases} \{ (\delta_m(q,1), n_i), (\delta m(q,0), n_{i+1}) \} & \text{if } a = 1 \\ \{ (\delta m(q,0), n_i), (\delta m(q,1), n_{i+1}) \} & \text{if } a = 0 \end{cases}$$

For $i = k$: $\delta_N((q, n_i), a) = \delta_M(q, a)$ with $a \in \{0, 1\}$

The following NFA accepts strings in $L_{\leq k}$. Looking at the transition function we see that if there is an input Say 1 that leads to State q in the DFA for L then according to our definition we would accept an input of 0 which would lead to q and then still have the remaining states as the original DFA. We are limited by the number of variations accepted because we only have k

copies of the original DFA. So if for instance $k = 6$ our NFA would be able to accept 6 variations in the input string where each variation would be accepted but cause our NFA to move down a copy of the DFA. So now to get to accepting states either the string continues through the DFA copy and matches the ~~input~~ remaing part of the input string exactly to the string being compared to from L or the NFA keeps on going down the copies of the DFA and accepting up to k variation where at the k_0 copy the remaining part of the input string must match exactly to the string from L for the remaing DFA states inorder for the input string to be accepted. And this argument can be used for any integer k.

5 2E 20 / 20

✓ - 0 pts **Correct formal definition of NFA and clear proof of correctness**

- 5 pts proof of correctness of NFA mingled with description of NFA and/or is incomplete, such as showed NFA accepts L<=k but doesn't show it doesn't accept other strings
- 10 pts no proof of correctness of NFA
- 10 pts Incorrect NFA construction
- 0 pts proof by induction correct and complete
- 3 pts proof of induction have minor mistakes
- 8 pts proof of induction vague or missing steps, e.g., inductive step uses (D) to extend without specific details
- 20 pts Incorrect NFA construction and proof idea
- 15 pts IDK
- 20 pts Ineligible: We were unable to follow the logic of the answer, or the answer is just way too long. In the future you might want to consider using IDK.
- 20 pts unreadable
- 5 pts NFA construction idea unclear

gradescope