CS 181 HW4 2021 CS181

CHARLES ZHANG

TOTAL POINTS

21 / 22

QUESTION 1

1 GNFA 2 / 2

✓ **- 0 pts** Correct

- 1 pts First Incorrect

- 1 pts Second Incorrect

QUESTION 2

2 Language of CFG 3 / 3

✓ **- 0 pts** Correct

- 1 pts Exactly 2 #'s, not exactly 1.

- 1 pts Exactly 2 #'s, not 2 or more.

QUESTION 3

3 Closure under Reversal 4 / 4

✓ **- 0 pts** Correct answer

- 2 pts Failed to construct a correct regular expression, or DFA/NFA that recognizes the $$A^R$$. Do not provide any construction process

- 1 pts Majority of the answer is correct, but lack of explanation how to construct the regular expression, or DFA/NFA(GNFA) for the reversed language. Noted that we do not accept a single diagram to illustrate the construction procedure without any explanation. We expected an **adequate** explanation of how to construct a general model, e.g., how you reverse the "path". And your solution should cover all of following cases:
- How to define the final states (and its transitions) of the model for reversed language
- How to define the start state and its transitions (noted that there is only one start state for a FA) of the model for the reversed language
- How to define the transitions of the model for the reversed language
- The original model could have multiple final states.

- 1 pts Majority of the answer is correct, but lack of explanation of why your constructed expression/DFA/NFA correctly recognize the $$A^R$$, and why A^R is a FSL.

- 4 pts Did not answer this problem.

QUESTION 4

4 CFG for Language 5 / 6

- 0 pts Correct

✓ **- 1 pts** Almost correct

- 2 pts Partialy Correct

- 4 pts Attempted

- 6 pts Not Attempted

QUESTION 5

5 Pumping Lemma for FSLs 7 / 7

✓ **+ 7 pts** Answer is correct or nearly correct.

+ 1 pts Appropriate string

+ 2 pts Use of constraints on xyz is effective

+ 1 pts Use of constraints is partially correct

+ 2 pts Show coverage of all case(s) is correct

+ 1 pts Show coverage is partially correct

+ 2 pts Logic in all case(s) is clear, complete, & correct

+ 1 pts Logic is partially clear, complete, & correct

- 0.5 pts Should clearly state that |xy| \leq p implies ...

+ 0 pts Cannot assume specific value for p

+ 0 pts Cannot assume specific values for x,y,z

+ 0 pts Your string can always be pumped and stay in the language.

+ 0 pts Your string is not in the language.

+ 0 pts You need to choose a specific string

+ 0 pts No answer.

CS 181 Homework 4

Charles Zhang, 305-413-659

April 26, 2021

Problem 1

Path 1: $q_{start} \longrightarrow ab^\star \longrightarrow q_1 \longrightarrow a^\star \longrightarrow q_2 \longrightarrow (aa)^\star \longrightarrow q_1 \longrightarrow ab \cup ba \longrightarrow q_{accept}$

- $ab^\star$ matches $\boldsymbol{a}aab$
- $a^\star$ matches $a\boldsymbol{a}ab$
- $(aa)^\star$ matches ϵ
- $ab \cup ba$ matches $aa\boldsymbol{ab}$

Path 2: $q_{start} \longrightarrow ab^\star \longrightarrow q_1 \longrightarrow a^\star \longrightarrow q_2 \longrightarrow b^\star \longrightarrow q_{accept}$

- $ab^\star$ matches $\boldsymbol{a}aab$
- $a^\star$ matches $a\boldsymbol{aa}b$
- $b^\star$ matches $aaa\boldsymbol{b}$

Problem 2

This CFG generates a language that has exactly two '#' symbols. Each of these '#' symbols can have 0 or more '0' symbols to its left and 0 or more '0' symbols to its right. The number of '0' symbols to the left and right of each '#' are independent of one another.

1 GNFA 2 / 2

- ✓ **- 0 pts** Correct
- **- 1 pts** First Incorrect
- **- 1 pts** Second Incorrect

CS 181 Homework 4

Charles Zhang, 305-413-659

April 26, 2021

Problem 1

Path 1: $q_{start} \longrightarrow ab^\star \longrightarrow q_1 \longrightarrow a^\star \longrightarrow q_2 \longrightarrow (aa)^\star \longrightarrow q_1 \longrightarrow ab \cup ba \longrightarrow q_{accept}$

- $ab^\star$ matches $\boldsymbol{a}aab$
- $a^\star$ matches $a\boldsymbol{a}ab$
- $(aa)^\star$ matches ϵ
- $ab \cup ba$ matches $aa\boldsymbol{ab}$

Path 2: $q_{start} \longrightarrow ab^\star \longrightarrow q_1 \longrightarrow a^\star \longrightarrow q_2 \longrightarrow b^\star \longrightarrow q_{accept}$

- $ab^\star$ matches $\boldsymbol{a}aab$
- $a^\star$ matches $a\boldsymbol{aa}b$
- $b^\star$ matches $aaa\boldsymbol{b}$

Problem 2

This CFG generates a language that has exactly two '#' symbols. Each of these '#' symbols can have 0 or more '0' symbols to its left and 0 or more '0' symbols to its right. The number of '0' symbols to the left and right of each '#' are independent of one another.

2 Language of CFG 3 / 3

✓ - **0 pts** Correct

- **1 pts** Exactly 2 #'s, not exactly 1.
- **1 pts** Exactly 2 #'s, not 2 or more.

Problem 3

Proof by construction:

- $\exists$ NFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts A
- Goal: construct a new NFA $M' \ni M'$ accepts A^R
- For some string $w = w_1 w_2 ... w_n$ in A, an accepting computation begins at q_0, using symbols $w_1 w_2 ... w_n$ and the transition function δ to transition to other states $q_i \in Q$ until the machine ends up in some $q_x \ni q_x \in F$
- Let $M' = (Q', \Sigma, \delta', q_0', F')$ such that:
  - $Q' \equiv Q \cup \{q_0'\}$
  - $\delta'(q_0', \epsilon) \equiv F$
  - $\delta'(j, x) \equiv \{i \mid \delta(i, x) = j\} \; \forall \; p, q \in Q, x \in \Sigma$
  - $F' \equiv \{q_0\}$
- Claim: M' accepts A^R

Justification:
The basic idea of modeling M' is that we use the same NFA as M, but traverse it backwards. This should allow us to essentially accept strings of the form $w \in A$ in reverse order, which is the definition of A^R, according to the problem statement. We do this by defining δ' such that, for any transition from state i to state j using symbol x in M, there is a corresponding transition from state j to state i using symbol x in M'. We then account for the fact that M may have multiple accept states by adding a ϵ transition from a new state (and start state) q_0' to each of the accept states of M. Finally, we designate the start state of M, q_0, as the only accept state of M', completing the idea that a string w' must traverse each of M's states in reverse to be accepted by M'.

3 Closure under Reversal **4 / 4**

✓ **- 0 pts Correct answer**

- 2 pts Failed to construct a correct regular expression, or DFA/NFA that recognizes the $$A^R$$. Do not provide any construction process

- 1 pts Majority of the answer is correct, but lack of explanation how to construct the regular expression, or DFA/NFA(GNFA) for the reversed language. Noted that we do not accept a single diagram to illustrate the construction procedure without any explanation. We expected an **adequate** explanation of how to construct a general model, e.g., how you reverse the "path". And your solution should cover all of following cases:
- How to define the final states (and its transitions) of the model for reversed language
- How to define the start state and its transitions (noted that there is only one start state for a FA) of the model for the reversed language
- How to define the transitions of the model for the reversed language
- The original model could have multiple final states.

- 1 pts Majority of the answer is correct, but lack of explanation of why your constructed expression/DFA/NFA correctly recognize the $$A^R$$, and why A^R is a FSL.

- 4 pts Did not answer this problem.

Problem 4

$G = (V, \Sigma, R, S)$, with $V = \{S, M, D, P, K, T\}$ and R = the rule set below.

$$S \longrightarrow M \mid D$$

$$M \longrightarrow aMa \mid bMb \mid cMc \mid \#K\#$$

$$D \longrightarrow P\#K$$

$$P \longrightarrow TTPT \mid \#$$

$$K \longrightarrow aK \mid bK \mid cK \mid \epsilon$$

$$T \longrightarrow a \mid b \mid c$$

Justification:
This CFG splits the CFL into two cases: languages that satisfy $z = x^R$ and languages that satisfy $|y| = 2|x|$. We handle the reverse case using the R rule, recursively adding matching symbols in the language to either side of R, modelling how x mirrors z. We then end this recursion by filling in the # symbols and K, which acts as a variable that represents $y \in \Sigma^\star$. We handle the "doubling" case by using the P rule, which recursively adds twice as many T variables (which are just a single terminal from the alphabet) to the left side of P as the right side. This models x being twice as long as y. P finishes recursion with the # separator. Finally, the rule is concluded using the D rule to combine P, the # separator, and the K rule to allow $z \in \Sigma^\star$. These two cases are mashed together in the starting rule S.

4 CFG for Language 5 / 6

- **- 0 pts** Correct
- ✓ **- 1 pts Almost correct**
- **- 2 pts** Partialy Correct
- **- 4 pts** Attempted
- **- 6 pts** Not Attempted

Problem 5

Proof (by contradiction):

- Assume L_5 is regular
- Let p be the pumping length given by the pumping lemma
- Let $w = 0^p 1 1 0^p$, noting that $w \in L_5$
- By Sipser's condition 1 of the pumping lemma, we know that w can be split into three parts such that $w = abc$ and for any $i \geq 0$, the string $w' = ab^i c$ is in L_5
- Sipser's condition 3 of the pumping lemma states that, when pumping w, it must be split such that $|ab| \leq p$
- Since w begins with p occurrences of 0, condition 3 guarantees that ab is made up entirely of 0s
- Sipser's condition 2 of the pumping lemma states that, when pumping w, it must be split such that $|b| \geq 1$
- Taken together, these conditions imply that b must contain at least one 0, and be made up entirely of 0s, therefore, $b = 0^t$, where $t \geq 1$
- We then pump the substring b using $i = 3$
- $w' = 0^{p+2t} 1 1 0^p$
- $|w'| = 2p + 2t + 2$
- In order for w' to satisfy L_5's condition that $|x| = |y|$, it must be true that $|x| = |y| = \frac{|w'|}{2} = \frac{2p+2t+2}{2} = p + t + 1$
- Since $t \geq 1$, it must follow that $p + t + 1 \leq p + 2t$
- Therefore, x must be made up of all 0s, as the first $p + 2t$ symbols of w' are 0s
- In other words, $\#(0, x) = |x|$
- For w' to be in L_5, it must also be true that $\#(0, y) = |x|$
- From the first condition of L_5, we know that $\#(0, y) = |y|$
- However, this is impossible, as the substring 11 must appear in y
- Thus, Sipser's condition 1 of the pumping lemma is not satisfied, and a contradiction has been found $\Rightarrow\!\Leftarrow$

5 Pumping Lemma for FSLs **7 / 7**

- ✓ **+ 7 pts** **Answer is correct or nearly correct.**
- **+ 1 pts** Appropriate string
- **+ 2 pts** Use of constraints on xyz is effective
- **+ 1 pts** Use of constraints is partially correct
- **+ 2 pts** Show coverage of all case(s) is correct
- **+ 1 pts** Show coverage is partially correct
- **+ 2 pts** Logic in all case(s) is clear, complete, & correct
- **+ 1 pts** Logic is partially clear, complete, & correct
- **- 0.5 pts** Should clearly state that |xy| \leq p implies ...
- **+ 0 pts** Cannot assume specific value for p
- **+ 0 pts** Cannot assume specific values for x,y,z
- **+ 0 pts** Your string can always be pumped and stay in the language.
- **+ 0 pts** Your string is not in the language.
- **+ 0 pts** You need to choose a specific string
- **+ 0 pts** No answer.