CS 181 HW4 2021 CS181

YIQIAO JIN

TOTAL POINTS

15 / 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 6 / 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 0 / 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.

Homework 4

Name: Yiqiao Jin
UID: 305107551

1

The string $aaab$ can be accepted by the following ways:

$$q_{start} \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{ab} q_{accept}$$

$$q_{start} \xrightarrow{a} q_1 \xrightarrow{aa} q_2 \xrightarrow{b} q_{accept}$$

2

The language represented by G_2 is any string formed by any number of 0's, as well as exactly two $\#$'s at any positions of the string.

3

Let the Finite State Language A be accepted by the **DFA** $D = (Q, \Sigma, \delta, q_0, F)$.

Here, Q is the set of States; Σ is the Alphabet; δ is the Transition Function; $q_0 \in Q$ is the start state; and $F \subseteq Q$ is the set of accept states.

We can design a new Finite Automaton $M = (Q, \Sigma, \delta', p_0, F')$ such that

- $F' = \{q_0\}$. This means the original start state q_0 in D is the new accept state in M
- δ' is the new transition function. For any symbol w, if $\delta(S_1, w) = S_2$ in D, then $\delta'(S_2, w) = S_1$ in M. On the state diagram, M is derived by reversing all the transition arrows of D.
- p_0 is the new start state of M with ϵ transition into all the accept state in D

For any string $w \in A$, there exists a path in $D : q_0 \rightarrow S_1 \rightarrow S_2 \rightarrow ... \rightarrow S_n \rightarrow S_a$, which accepts $w = w^1 w^2 ... w^n$

By our definition, for any $w^R \in A^R$ there must exist a path $p_0 \rightarrow S_a \rightarrow S_n \rightarrow ... \rightarrow S_2 \rightarrow S_1 \rightarrow q_0$ which accepts $w^R = w^n ... w^2 w^1$.

If a language can be accepted by a finite automaton, then the language is regular. Thus, w^R is accepted by M and is also regular.

1 GNFA **2 / 2**

✓ **- 0 pts** **Correct**

- 1 pts First Incorrect

- 1 pts Second Incorrect

Homework 4

Name: Yiqiao Jin
UID: 305107551

1

The string $aaab$ can be accepted by the following ways:

$$q_{start} \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{ab} q_{accept}$$

$$q_{start} \xrightarrow{a} q_1 \xrightarrow{aa} q_2 \xrightarrow{b} q_{accept}$$

2

The language represented by G_2 is any string formed by any number of 0's, as well as exactly two $\#$'s at any positions of the string.

3

Let the Finite State Language A be accepted by the **DFA** $D = (Q, \Sigma, \delta, q_0, F)$.

Here, Q is the set of States; Σ is the Alphabet; δ is the Transition Function; $q_0 \in Q$ is the start state; and $F \subseteq Q$ is the set of accept states.

We can design a new Finite Automaton $M = (Q, \Sigma, \delta', p_0, F')$ such that

- $F' = \{q_0\}$. This means the original start state q_0 in D is the new accept state in M
- δ' is the new transition function. For any symbol w, if $\delta(S_1, w) = S_2$ in D, then $\delta'(S_2, w) = S_1$ in M. On the state diagram, M is derived by reversing all the transition arrows of D.
- p_0 is the new start state of M with ϵ transition into all the accept state in D

For any string $w \in A$, there exists a path in $D : q_0 \to S_1 \to S_2 \to ... \to S_n \to S_a$, which accepts $w = w^1 w^2 ... w^n$

By our definition, for any $w^R \in A^R$ there must exist a path $p_0 \to S_a \to S_n \to ... \to S_2 \to S_1 \to q_0$ which accepts $w^R = w^n ... w^2 w^1$.

If a language can be accepted by a finite automaton, then the language is regular. Thus, w^R is accepted by M and is also regular.

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.

Homework 4

Name: Yiqiao Jin
UID: 305107551

1

The string $aaab$ can be accepted by the following ways:

$$q_{start} \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{ab} q_{accept}$$

$$q_{start} \xrightarrow{a} q_1 \xrightarrow{aa} q_2 \xrightarrow{b} q_{accept}$$

2

The language represented by G_2 is any string formed by any number of 0's, as well as exactly two $\#$'s at any positions of the string.

3

Let the Finite State Language A be accepted by the **DFA** $D = (Q, \Sigma, \delta, q_0, F)$.

Here, Q is the set of States; Σ is the Alphabet; δ is the Transition Function; $q_0 \in Q$ is the start state; and $F \subseteq Q$ is the set of accept states.

We can design a new Finite Automaton $M = (Q, \Sigma, \delta', p_0, F')$ such that

- $F' = \{q_0\}$. This means the original start state q_0 in D is the new accept state in M
- δ' is the new transition function. For any symbol w, if $\delta(S_1, w) = S_2$ in D, then $\delta'(S_2, w) = S_1$ in M. On the state diagram, M is derived by reversing all the transition arrows of D.
- p_0 is the new start state of M with ϵ transition into all the accept state in D

For any string $w \in A$, there exists a path in $D : q_0 \to S_1 \to S_2 \to ... \to S_n \to S_a$, which accepts $w = w^1 w^2 ... w^n$

By our definition, for any $w^R \in A^R$ there must exist a path $p_0 \to S_a \to S_n \to ... \to S_2 \to S_1 \to q_0$ which accepts $w^R = w^n ... w^2 w^1$.

If a language can be accepted by a finite automaton, then the language is regular. Thus, w^R is accepted by M and is also regular.

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.

4

A Context-Free Grammar G_4 that generates this language L_4 is:

$\mathbf{S} \rightarrow \mathbf{A}|\mathbf{CD}$

$\mathbf{A} \rightarrow a\mathbf{A}a|b\mathbf{A}b|\#\mathbf{B}\#$

$\mathbf{B} \rightarrow \mathbf{BE}|\varepsilon$

$\mathbf{C} \rightarrow \mathbf{ECEE}|\#$

$\mathbf{D} \rightarrow \mathbf{D}a|\mathbf{D}b|\#$

$\mathbf{E} \rightarrow a|b$

There are two kind of input strings we can have: $z = x^R$, or $|y| = 2|x|$

The first case $z = x^R$ is satisfied by $\mathbf{A}$, which generates z and x^R by creating two identical symbols at the beginning and the end of the new variable $\mathbf{A}$ each time. After generating x and z, $\mathbf{A}$ can also generate the string $y \in \{a, b\}^*$ enclosed in a pair of $\#$, which has a length in $[0, \infty)$. This is done by $\mathbf{B}$

The second case $|y| = 2|x|$ is satisfied by letting $\mathbf{C}$ generates one variable $\mathbf{E}$ at its beginning and two variables $\mathbf{E}$ at its ending. $\mathbf{C}$ can also generate the terminal $\#$, which is done when both x and y are fully generated. Then, $\mathbf{D}$ generates the rest of z.

Note that the 4th rule is the same as $\mathbf{D} \rightarrow \mathbf{DE}|\#$

5

We prove that there exists some string $s \in L_5$ that cannot satisfy the pumping lemma

Let s be a string in the form $s = xy = 0^n 1^{2n} 0^n$, where $x = 0^n 1^n$ and $y = 1^n 0^n$. s satisfies that $|x| = |y|$ and $\#(0, x) = \#(0, y) = n$. So $s \in L_5$

Suppose L_5 were an FSL. We can apply the pumping lemma. Let p be the pumping length in the pumping lemma.

Since $s \in L_5$, and $|s| \geq p$, there exist some substrings a, b, c such that s can be written as $s = abc$, where:

- $|ab| \leq p$
- $|b| \geq 1$
- for all $i \geq 0$, $ab^i c \in L_5$.

4 CFG for Language 6 / 6

✓ **- 0 pts** **Correct**

- 1 pts Almost correct

- 2 pts Partialy Correct

- 4 pts Attempted

- 6 pts Not Attempted

4

A Context-Free Grammar G_4 that generates this language L_4 is:

$\mathbf{S} \rightarrow \mathbf{A}|\mathbf{CD}$

$\mathbf{A} \rightarrow a\mathbf{A}a|b\mathbf{A}b|\#\mathbf{B}\#$

$\mathbf{B} \rightarrow \mathbf{BE}|\varepsilon$

$\mathbf{C} \rightarrow \mathbf{ECEE}|\#$

$\mathbf{D} \rightarrow \mathbf{D}a|\mathbf{D}b|\#$

$\mathbf{E} \rightarrow a|b$

There are two kind of input strings we can have: $z = x^R$, or $|y| = 2|x|$

The first case $z = x^R$ is satisfied by $\mathbf{A}$, which generates z and x^R by creating two identical symbols at the beginning and the end of the new variable $\mathbf{A}$ each time. After generating x and z, $\mathbf{A}$ can also generate the string $y \in \{a, b\}^*$ enclosed in a pair of $\#$, which has a length in $[0, \infty)$. This is done by $\mathbf{B}$

The second case $|y| = 2|x|$ is satisfied by letting $\mathbf{C}$ generates one variable $\mathbf{E}$ at its beginning and two variables $\mathbf{E}$ at its ending. $\mathbf{C}$ can also generate the terminal $\#$, which is done when both x and y are fully generated. Then, $\mathbf{D}$ generates the rest of z.

Note that the 4th rule is the same as $\mathbf{D} \rightarrow \mathbf{DE}|\#$

5

We prove that there exists some string $s \in L_5$ that cannot satisfy the pumping lemma

Let s be a string in the form $s = xy = 0^n 1^{2n} 0^n$, where $x = 0^n 1^n$ and $y = 1^n 0^n$. s satisfies that $|x| = |y|$ and $\#(0, x) = \#(0, y) = n$. So $s \in L_5$

Suppose L_5 were an FSL. We can apply the pumping lemma. Let p be the pumping length in the pumping lemma.

Since $s \in L_5$, and $|s| \geq p$, there exist some substrings a, b, c such that s can be written as $s = abc$, where:

- $|ab| \leq p$
- $|b| \geq 1$
- for all $i \geq 0$, $ab^i c \in L_5$.

1) b only contain 1's

The assumption implies $p \leq 2n$. Without loss of generality, we assume b is at the center of s, since either way, the pumped string $s' = ab^ic = 0^n1^{2n+p(i-1)}0^n$. The 1's in the center of the newly pumped string will always remain continuous.

Note that p must be even ($p/2$ is an integer) because if p is odd, the length $|s'|$ will be odd for some $s' = ab^ic$, which makes $|x| \neq |y|$ and invalidates s' as a member of L_5

Then $s = 0^n1^{2n}0^n = \underbrace{0^n1^{n-p/2}}_{a}\underbrace{1^p}_{b}\underbrace{1^{n-p/2}0^n}_{c}$

However, if we pump up, $s' = ab^2c = \underbrace{0^n1^{n-p/2}}_{a}\underbrace{1^{2p}}_{b^2}\underbrace{1^{n-p/2}0^n}_{c} = 0^n1^{2n+p}0^n$. Now $s' \notin L_5$, which invalidates the 3rd condition of the pumping lemma

2) b only contain 0's

The assumption implies $p \leq n$. Assume b is in the starting 0^n. Similarly, $s = 0^n1^{2n}0^n = \underbrace{0^k}_{a}\underbrace{0^p}_{b}\underbrace{0^{n-k-p}1^{2n}0^n}_{c}$ for some $k \geq 0$. Pumping down will generate the string s' such that

$$s' = ac = \underbrace{0^k}_{a}\underbrace{0^{n-k-p}1^{2n}0^n}_{c} = 0^{n-p}1^{2n}0^n \notin L_5$$

The same result can be proved when b is in the trailing 0^n.

3) p contains both 0's and 1's

As long as p contains both 0's and 1's, pumping up the string s will insert 0's in-between the consecutive sequence of 1^{2n} within s, which invalidate s' as a member of L_5.

In all, there exists string $s \in L_5$ that cannot satisfy the pumping lemma. Therefore, we prove that L_5 is NOT an FSL.

5 Pumping Lemma for FSLs **0 / 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.