CS 181 HW3 2021 CS181

CHARLES ZHANG

TOTAL POINTS

28 / 28

QUESTION 1

1 Pumping Lemma 7 / 7

✓ **+ 7 pts** Correct or nearly correct

+ 1 pts Appropriate "s"

+ 2 pts Effective use of constraints on "xyz"

+ 2 pts Show coverage of all cases of "xyz"

- 0.5 pts should give more complete justification that your proof covers all cases of xyz

+ 2 pts Sound logic in every case

- 0.5 pts Should clearly state that |xy| $\leq$ p means xy is all a's, not just mention that |xy| $\leq$ p.

- 0.5 pts minor error in logic

- 1 pts should give more complete justification logic

- 2 pts You cannot assign/assume any particular value for x, y, or z.

- 1 pts Cannot assume p is even

+ 0 pts No answer

QUESTION 2

2 Regular Expression 4 / 4

✓ **- 0 pts** Correct

- 1 pts Almost correct

- 3 pts Not correct

- 4 pts Not Attempted

QUESTION 3

3 NFA 5 / 5

✓ **- 0 pts** Correct

- 1 pts Minor mistakes, your NFA cannot accept some of the strings in the language. We use following strings to test your NFA.

- ababccbcccc
- ababcccc
- abababacccc
- ababbcccc
- ababbacccc
- babaacccc
- babaabcccc
- babaaccccabba
- ababbaccccbaab

- 1 pts Minor mistakes, your NFA will accept some strings that are not in the language, for example,

- ababccc
- babaabab
- ababbaba
- ccccbaba
- ccccabab
- ccccaababa
- ccccbbabab

- 1 pts Does not effectively use the nondeterminism in the NFA

- 1 pts Invalid NFA, or your NFA does not satisfy the definition of NFA, or do not explicitly specify the final states, or forget to specify the transition for some states.

- 5 pts Totally wrong, or you did not answer this problem at all.

QUESTION 4

Parse Trees & Derivations 4 pts

4.1 a Parse Tree 1 / 1

✓ **- 0 pts** Correct

- 0.5 pts Small mistake

- 1 pts Incorrect

4.2 b Left-most Derivation 1 / 1

✓ **- 0 pts** Correct

- 0.5 pts Missing Steps

- **0.5 pts** Not Left-most
- **1 pts** Incorrect

4.3 c Parse Tree **1 / 1**

✓ **- 0 pts** Correct

- **0.5 pts** Small Mistake
- **1 pts** Incorrect

4.4 d Left-most Derivation **1 / 1**

✓ **- 0 pts** Correct

- **0.5 pts** Missing Steps
- **0.5 pts** Not Left-most
- **1 pts** Incorrect

QUESTION 5

5 CFG; **4 / 4**

✓ **- 0 pts** Correct

- **1 pts** Can not express arbitrary number of begin end blocks beside each other, e.g. bbs;e;bs;e;bs;e;e;
- **2 pts** Begin and end blocks don't have to line up. e.g. bbs;e; or bs;e;e; is generated even though it shouldn't be.
- **1 pts** No specified start variable
- **1 pts** Extra semicolons generated (e.g. bs;;bs;e;e; or b;s;e;)
- **1 pts** Can not express multiple statements beside each other (e.g. bs;s;s;e;)
- **1 pts** Missing semicolons on s (e.g. bse; is generated)
- **1 pts** Can not express single statement e.g. bs;e;
- **1 pts** Can't do some orders of statements and blocks e.g. bbs;e;s;e; or bs;bs;e;e;
- **1 pts** Can't have arbitrary nestings next to each other (like bbbs;e;e;bbs;e;e;e;
- **1 pts** Doesn't necessarily have an outside begin end pair (e.g. generates s; or b or nothing at all, or bs;e;bs;e;)
- **1 pts** There isn't necessarily an outer be pair (e.g. bs;e;bs;e; can be generated)
- **0 pts** Click here to replace this description.

QUESTION 6

6 CFG, **4 / 4**

✓ **- 0 pts** Correct

- **1 pts** Adjacent begin statements either can't exist (can't generate bbse,bsee) or can be missing commas between (e.g. bbsebsee)
- **1 pts** Can't generate arbitrarily many adjacent (or nested) begin/ends (e.g. bbse,bse,bsee or bbsee or bs,bse,bsee)
- **2 pts** zBegin/end not guaranteed to match (e.g. could generate bsee or bssee)
- **1 pts** No specified start variable
- **1 pts** Unnecessary semicolons
- **1 pts** Doesn't necessarily generate outside begin end pair (e.g. bsebse or bse,bse or s or s, or epsilon)
- **1 pts** Can't generate certain orders of statements, for example bbse,se or bs,bsee
- **1 pts** begin/end statements can be empty (e.g. generates be)
- **1 pts** Could be missing commas (e.g. this can generate bsse or bss,se or bbsese)
- **1 pts** Can't generate arbitrarily many s's (e.g. bs,s,se)
- **1 pts** Can generate extra commas (e.g. bse, or bs,e or b,,,,se)
- **1 pts** Can't do arbitrary nesting of begin/end statements (e.g. bbbseee

QUESTION 7

7 Postponed to next weeK: GNFA **0 / 0**

✓ **- 0 pts** Correct

CS 181 Homework 3

Charles Zhang, 305-413-659

April 17, 2021

Problem 1

Proof (by contradiction):

- Assume L is regular.
- Let p be the pumping length given by the pumping lemma.
- Let s be the string $a^{2p}b^p$.
- With s being a member of L and having length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$, the string xy^iz is in L.
- Sipser's condition 3 of the pumping lemma states that when pumping s, it must be divided so that $|xy| \leq p$.
- Since s begins with $2p$ occurrences of a, this means that xy must be made up of entirely as, otherwise $|xy| \geq p$.
- Since xy is made up entirely of as, it logically follows that y must be made up entirely of as.
- Since the language is specified as $a^{(2n)}b^n$, where n must represent the same number in both occurrences, we can conclude that a string w must contain exactly twice as many as as bs for it to be a member of L.
- Since y is guaranteed to be made up of all as, a string s' that is pumped such that $i = 2$ will not be a member of the language, as the number of as will increase, but the number of bs will not, so it can be said that the number of occurrences of b is not equal to exactly 2 times the number of occurrences of a.
- Therefore, the string s cannot be pumped, which contradicts the original assumption that L is regular by the pumping lemma.$\Rightarrow\Leftarrow$

1 Pumping Lemma 7 / 7

✓ **+ 7 pts** **Correct or nearly correct**

+ 1 pts Appropriate "s"

+ 2 pts Effective use of constraints on "xyz"

+ 2 pts Show coverage of all cases of "xyz"

- 0.5 pts should give more complete justification that your proof covers all cases of xyz

+ 2 pts Sound logic in every case

- 0.5 pts Should clearly state that |xy| $\leq$ p means xy is all a's, not just mention that |xy| $\leq$ p.

- 0.5 pts minor error in logic

- 1 pts should give more complete justification logic

- 2 pts You cannot assign/assume any particular value for x, y, or z.

- 1 pts Cannot assume p is even

+ 0 pts No answer

Problem 2

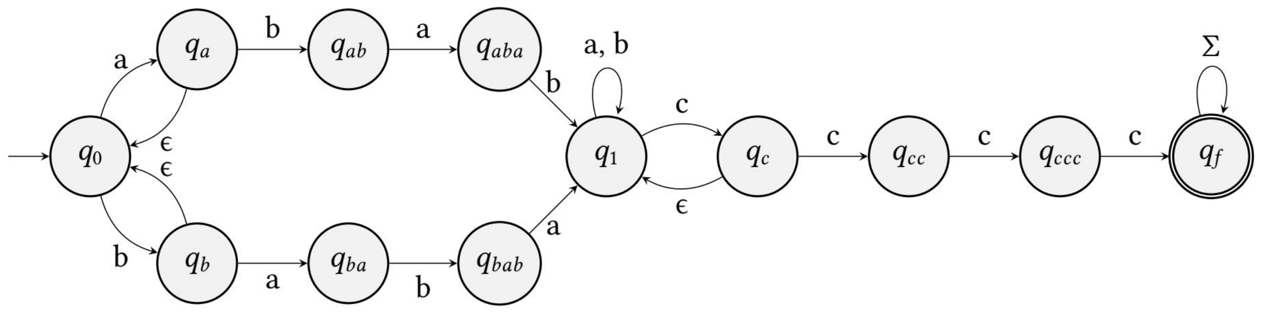
$$(a(a \cup b \cup c)^*(b \cup c)) \cup (b(a \cup b \cup c)^*(a \cup c)) \cup (c(a \cup b \cup c)^*(a \cup b))$$

This regular expression splits L_2 into three parts: strings that begin with a, strings that begin with b, and strings that begin with c. In each of these parts, it takes in the starting symbol S, 0 or more occurrences of any symbol in Σ, followed by an occurrence of E, where $E \in \Sigma - S$. The union of these three cases is then used to form the regular expression. This ensures both that the starting and ending symbol cannot match and that the string has a length greater than 1.

2 Regular Expression 4 / 4

- ✓ **- 0 pts** Correct
- **- 1 pts** Almost correct
- **- 3 pts** Not correct
- **- 4 pts** Not Attempted

Problem 3



This state diagram essentially splits L_3 into two parts: the part that detects the substrings *abab* and *baba*, and the part that detects the substring *cccc*. The part on the left detects either *abab* or *baba*, with ϵ-edges returning from q_a and q_b to q_0. These ϵ-edges allow us to use nondeterminism to continually "guess and check" where the *abab* or *baba* appear. This same logic is then repeated for the right half of the state diagram, but checking the substring *cccc* instead. Since this substring can occur anytime after one of *abab* or *baba* is found, we use another ϵ-edge to return to q_1 in order to guess and check where the *cccc* starts. Once *cccc* is found, the rest of the string doesn't matter, as all requirements for the string being in the language have been met, so all following inputs will continuously loop into the accept state q_f.

3 NFA 5 / 5

✓ **- 0 pts** **Correct**

- 1 pts Minor mistakes, your NFA cannot accept some of the strings in the language. We use following strings to test your NFA.

- ababccbcccc
- ababcccc
- ababacccc
- ababbcccc
- ababbacccc
- babaacccc
- babaabcccc
- babaaccccabba
- ababbaccccbaab

- 1 pts Minor mistakes, your NFA will accept some strings that are not in the language, for example,

- ababccc
- babaabab
- ababbaba
- ccccbaba
- ccccabab
- ccccaababa
- ccccbbabab

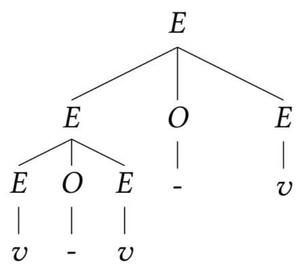
- 1 pts Does not effectively use the nondeterminism in the NFA

- 1 pts Invalid NFA, or your NFA does not satisfy the definition of NFA, or do not explicitly specify the final states, or forget to specify the transition for some states.

- 5 pts Totally wrong, or you did not answer this problem at all.

Problem 4

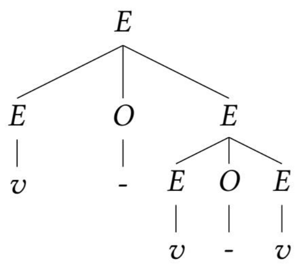
a)



b)

$\underline{E}$
$\underline{E}OE$
$\underline{E}OEOE$
$v\underline{O}EOE$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

c)



d)

$\underline{E}$
$\underline{E}OE$
$v\underline{O}E$
$v - \underline{E}$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

4.1 a Parse Tree 1 / 1

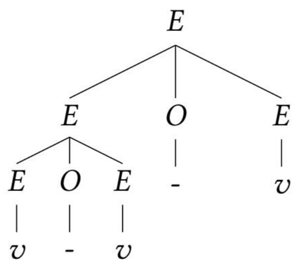
✓ **- 0 pts** Correct

- 0.5 pts Small mistake

- 1 pts Incorrect

Problem 4

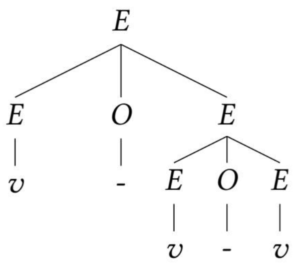
a)



b)

$\underline{E}$
$\underline{E}OE$
$\underline{E}OEOE$
$v\underline{O}EOE$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

c)



d)

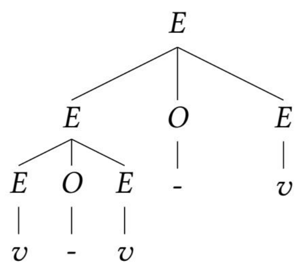
$\underline{E}$
$\underline{E}OE$
$v\underline{O}E$
$v - \underline{E}$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

4.2 b Left-most Derivation **1 / 1**

- ✓ **- 0 pts** **Correct**
- **- 0.5 pts** Missing Steps
- **- 0.5 pts** Not Left-most
- **- 1 pts** Incorrect

Problem 4

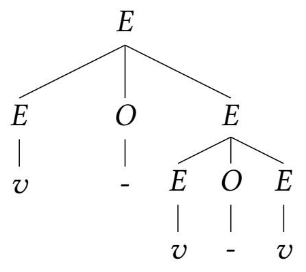
a)



b)

$\underline{E}$
$\underline{E}OE$
$\underline{E}OEOE$
$v\underline{O}EOE$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

c)



d)

$\underline{E}$
$\underline{E}OE$
$v\underline{O}E$
$v - \underline{E}$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

4.3 c Parse Tree 1 / 1

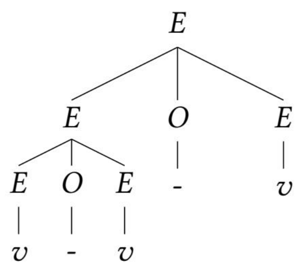
✓ - **0 pts** Correct

- **0.5 pts** Small Mistake

- **1 pts** Incorrect

Problem 4

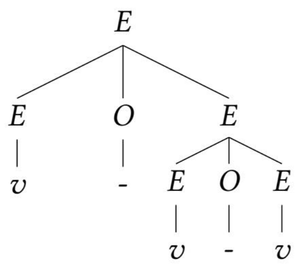
a)



b)

$\underline{E}$
$\underline{E}OE$
$\underline{E}OEOE$
$v\underline{O}EOE$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

c)



d)

$\underline{E}$
$\underline{E}OE$
$v\underline{O}E$
$v - \underline{E}$
$v - \underline{E}OE$
$v - v\underline{O}E$
$v - v - \underline{E}$
$v - v - v$

4.4 d Left-most Derivation **1 / 1**

✓ **- 0 pts** **Correct**

- 0.5 pts Missing Steps

- 0.5 pts Not Left-most

- 1 pts Incorrect

Problem 5

$$A \longrightarrow bBe;$$
$$B \longrightarrow s; C \mid bBe; C$$
$$C \longrightarrow s; C \mid bBe; C \mid \epsilon$$

Problem 6

$$A \longrightarrow bBe$$
$$B \longrightarrow sC \mid bBeC$$
$$C \longrightarrow , sC \mid , bBeC \mid \epsilon$$

5 CFG; **4 / 4**

✓ **- 0 pts** Correct

- 1 pts Can not express arbitrary number of begin end blocks beside each other, e.g. bbs;e;bs;e;bs;e;e;

- 2 pts Begin and end blocks don't have to line up. e.g. bbs;e; or bs;e;e; is generated even though it shouldn't be.

- 1 pts No specified start variable

- 1 pts Extra semicolons generated (e.g. bs;;bs;e;e; or b;s;e;)

- 1 pts Can not express multiple statements beside each other (e.g. bs;s;s;e;)

- 1 pts Missing semicolons on s (e.g. bse; is generated)

- 1 pts Can not express single statement e.g. bs;e;

- 1 pts Can't do some orders of statements and blocks e.g. bbs;e;s;e; or bs;bs;e;e;

- 1 pts Can't have arbitrary nestings next to each other (like bbbs;e;e;bbs;e;e;e;

- 1 pts Doesn't necessarily have an outside begin end pair (e.g. generates s; or b or nothing at all, or bs;e;bs;e;)

- 1 pts There isn't necessarily an outer be pair (e.g. bs;e;bs;e; can be generated)

- 0 pts Click here to replace this description.

Problem 5

$$A \longrightarrow bBe;$$
$$B \longrightarrow s;C \mid bBe;C$$
$$C \longrightarrow s;C \mid bBe;C \mid \epsilon$$

Problem 6

$$A \longrightarrow bBe$$
$$B \longrightarrow sC \mid bBeC$$
$$C \longrightarrow ,sC \mid ,bBeC \mid \epsilon$$

6 CFG, **4 / 4**

✓ **- 0 pts** Correct

- 1 pts Adjacent begin statements either can't exist (can't generate bbse,bsee) or can be missing commas between (e.g. bbsebsee)

- 1 pts Can't generate arbitrarily many adjacent (or nested) begin/ends (e.g. bbse,bse,bsee or bbsee or bs,bse,bsee)

- 2 pts zBegin/end not guaranteed to match (e.g. could generate bsee or bssee)

- 1 pts No specified start variable

- 1 pts Unnecessary semicolons

- 1 pts Doesn't necessarily generate outside begin end pair (e.g. bsebse or bse,bse or s or s, or epsilon)

- 1 pts Can't generate certain orders of statements, for example bbse,se or bs,bsee

- 1 pts begin/end statements can be empty (e.g. generates be)

- 1 pts Could be missing commas (e.g. this can generate bsse or bss,se or bbsese)

- 1 pts Can't generate arbitrarily many s's (e.g. bs,s,se)

- 1 pts Can generate extra commas (e.g. bse, or bs,e or b,,,,se)

- 1 pts Can't do arbitrary nesting of begin/end statements (e.g. bbbseee

Problem 5

$$A \longrightarrow bBe;$$
$$B \longrightarrow s; C \mid bBe; C$$
$$C \longrightarrow s; C \mid bBe; C \mid \epsilon$$

Problem 6

$$A \longrightarrow bBe$$
$$B \longrightarrow sC \mid bBeC$$
$$C \longrightarrow, sC \mid , bBeC \mid \epsilon$$

7 Postponed to next weeK: GNFA **0 / 0**

✓ **- 0 pts** **Correct**