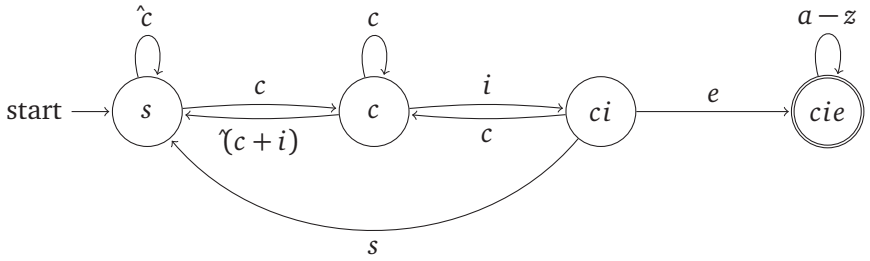
CS/ECE 374 B Fall 2019 Jiashuo Tong (jtong8)
Homework 1 Problem 1

Solution:

(a) The strings that violate the principle "i before e except after c" are strings that contain either cie or $\hat{}cei$. Thus, the regular expression for such strings can be written as $(a—z)^*(cie+\hat{}cei)(a—z)^*$.

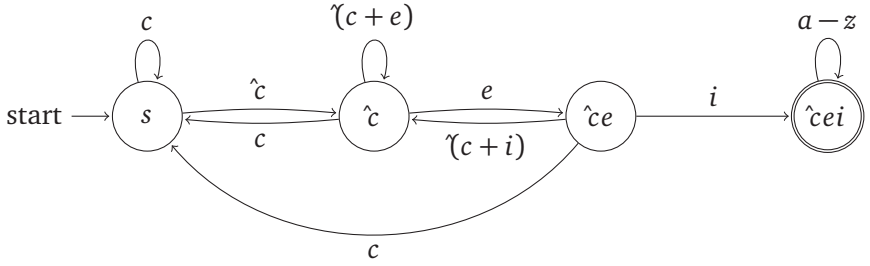
(b) First, we draw graphical representations for each of the two cases that violate the principle

1. The graph for DFA of strings that contain cie is



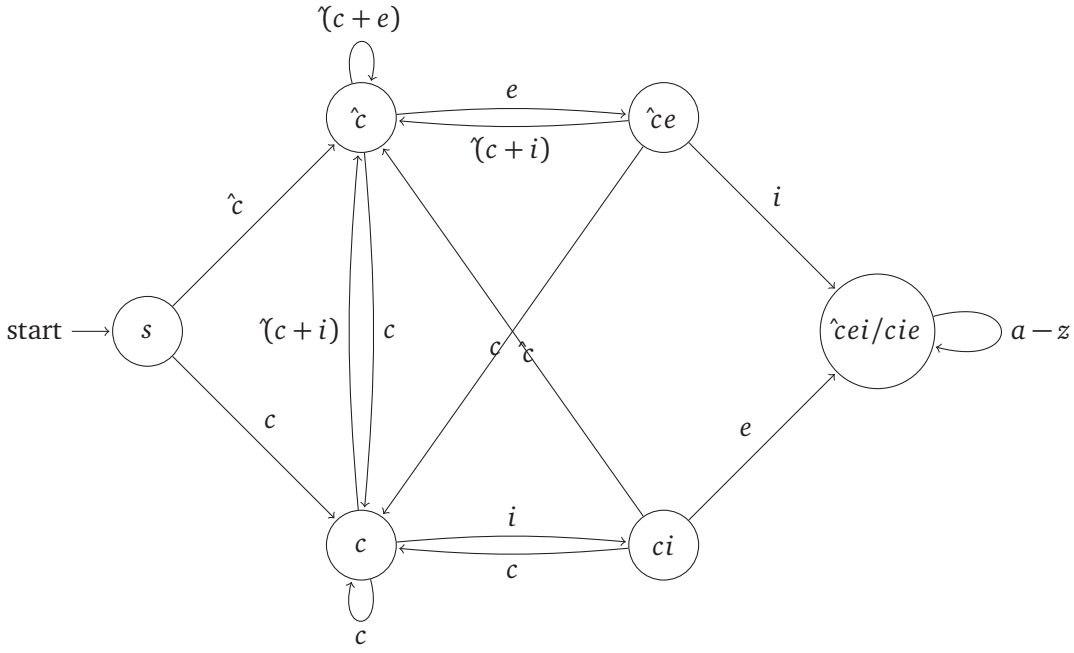
- s: strings end with $\hat{}c$ but not ci
- c: strings end with c
- ci: strings end with ci
- cie: strings contain either cie

2. The graph for DFA of strings that contain $\hat{}cei$ is



- s: strings are empty
- $\hat{}c$: strings end with $\hat{}c$
- $\hat{}ce$: strings end with $\hat{}ce$
- $\hat{}cei$: strings contain either $\hat{}cei$

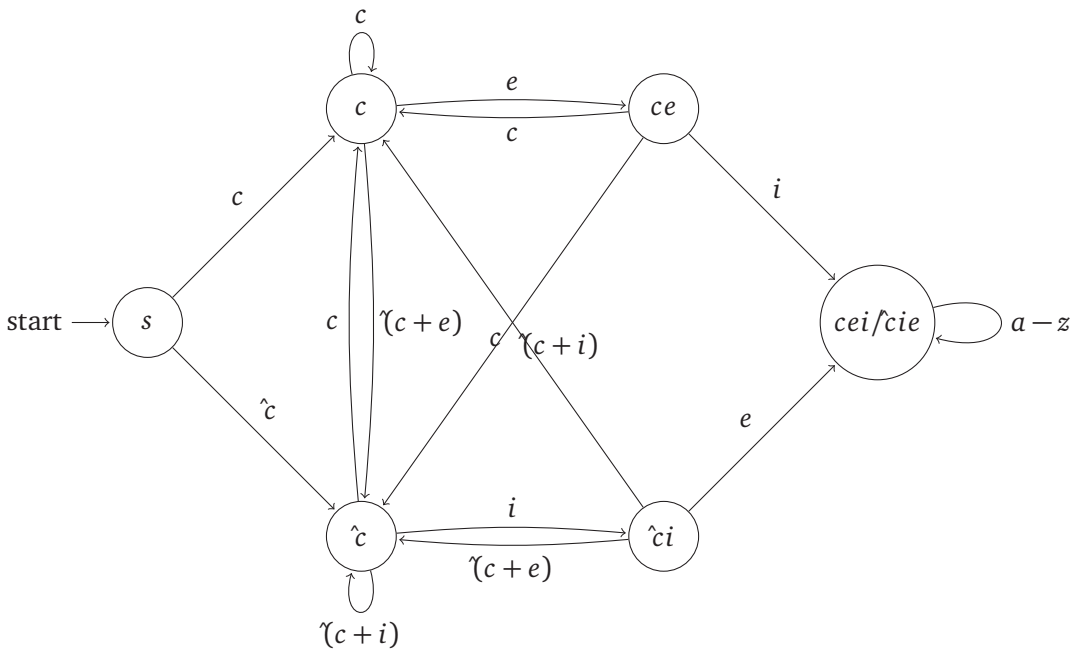
Next, we apply DFA profuct construction on the 2 cases, delete the states that do not exist, and finally obtain



- s: strings end with c
- $\hat{c}$: strings end with $\hat{c}$
- $\hat{c}e$: strings end with $\hat{c}e$
- c: strings end with c
- ci: strings end with ci
- $\hat{c}ei/cie$: strings contain either $\hat{c}ei$ or cie

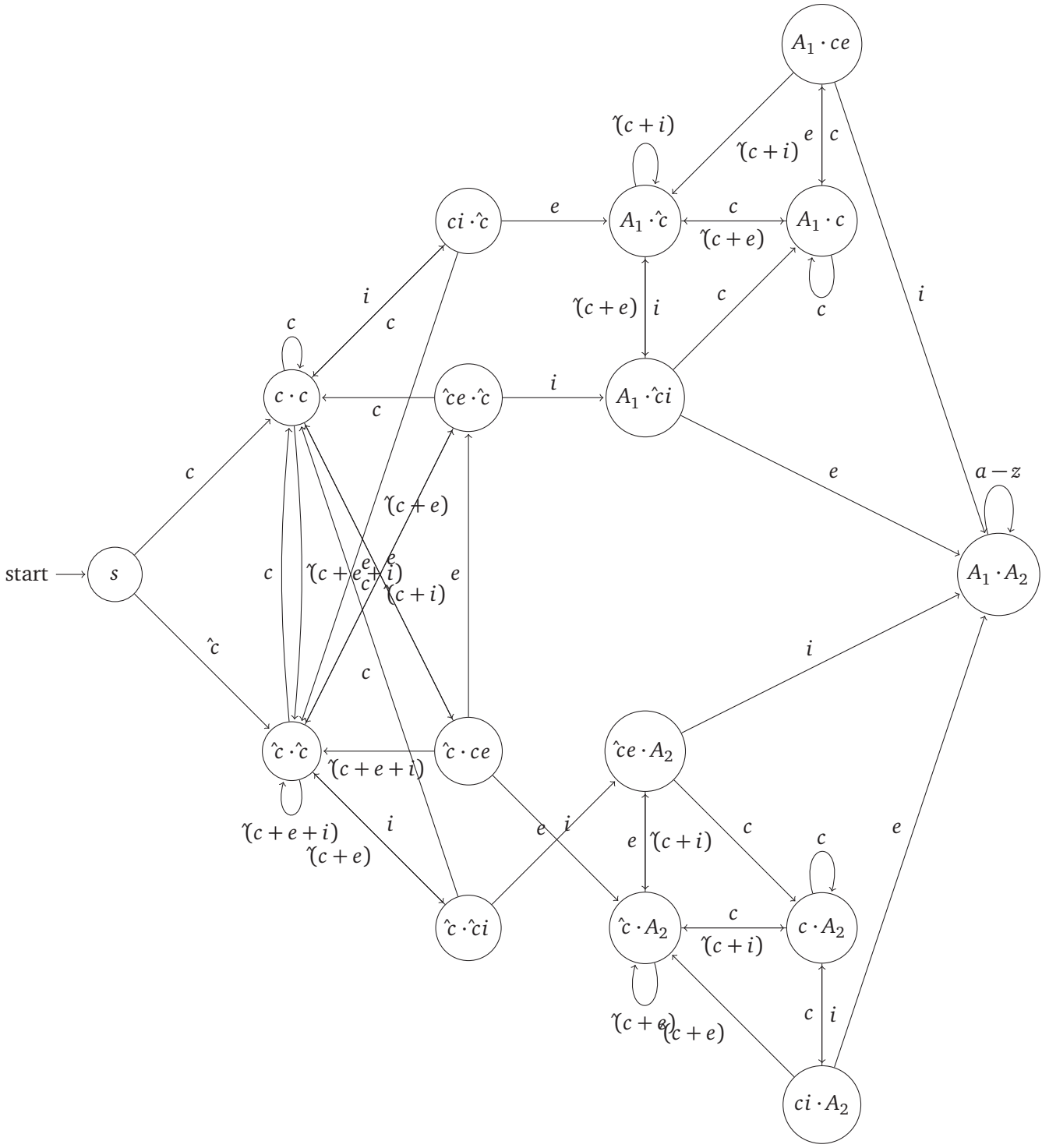
Why my DFA is correct? My DFA is correct because it is product constructed from the DFA that contains cie and the DFA that contains $\hat{c}ei$. Those are the only two sublanguages for a language that breaks the rule "i before e except after c".

(c) Similar to the DFA that breaks the rule, DFA that follows the rule is represented as



- s: strings are empty
- c: strings end with c
- ce: strings end with ce
- $\hat{c}$: strings end with $\hat{c}$
- $\hat{c}i$: strings end with $\hat{c}i$
- $\hat{c}ie/cei$: strings contain either $\hat{c}ie$ or cei

Product construction using DFA that breaks the rule and DFA that follows the rule gives the following DFA (graph representation):



- s: strings are empty
- $c \cdot c$: strings that are in states c for both DFA's
- $\hat{c} \cdot \hat{c}$: strings that are in states $\hat{c}$ for both DFA's
- $ci \cdot \hat{c}$: strings that are in states ci for 1st DFA and $\hat{c}$ for 2nd DFA
- $\hat{c}e \cdot \hat{c}$: strings that are in states $\hat{c}e$ for 1st DFA and c for 2nd DFA

- $\hat{c} \cdot ce$: strings that are in states $\hat{c}$ for 1st DFA and ce for 2nd DFA
- $\hat{c} \cdot \hat{c}i$: strings that are in states $\hat{c}$ for 1st DFA and $\hat{c}i$ for 2nd DFA
- $A_1 \cdot \hat{c}$: strings that are in states A_1 (breaking the rule) for 1st DFA and $\hat{c}$ for 2nd DFA
- $A_1 \cdot \hat{c}i$: strings that are in states A_1 (breaking the rule) for 1st DFA and $\hat{c}i$ for 2nd DFA
- $\hat{c}e \cdot A_2$: strings that are in states $\hat{c}e$ for 1st DFA and A_2 (following the rule) for 2nd DFA
- $\hat{c} \cdot A_2$: strings that are in states $\hat{c}$ for 1st DFA and A_2 (following the rule) for 2nd DFA
- $A_1 \cdot ce$: strings that are in states A_1 (breaking the rule) for 1st DFA and ce for 2nd DFA
- $A_1 \cdot c$: strings that are in states A_1 (breaking the rule) for 1st DFA and c for 2nd DFA
- $c \cdot A_2$: strings that are in states c for 1st DFA and A_2 (following the rule) for 2nd DFA
- $ci \cdot A_2$: strings that are in states ci for 1st DFA and A_2 (following the rule) for 2nd DFA
- $A_1 \cdot A_2$: strings that are in states A_1 (breaking the rule) for 1st DFA and A_2 (following the rule) for 2nd DFA

Why my DFA is correct? My DFA is correct because it is product constructed from the DFA that contains substrings breaking the rule and the DFA that contains substrings following the rule. Some states that do not exist (such as $c \cdot ce$) are deleted from the graph. **The lines and notations between the second and third layers are a little bit messy, but everything essential is included in the graph.**

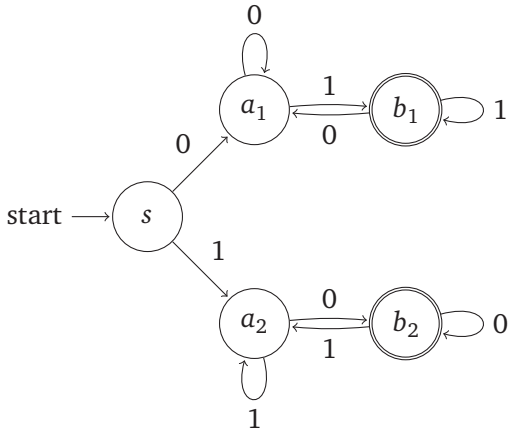
■

CS/ECE 374 B Fall 2019 Jiashuo Tong (jtong8)

Homework 1 Problem 2

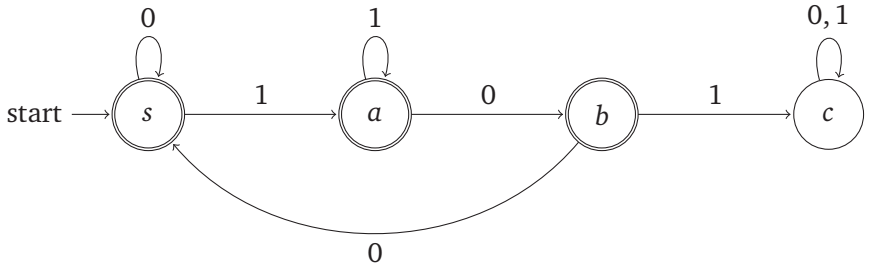
Solution:

(a) Regular Expression: $0(0+1)^*1+1(0+1)^*0$



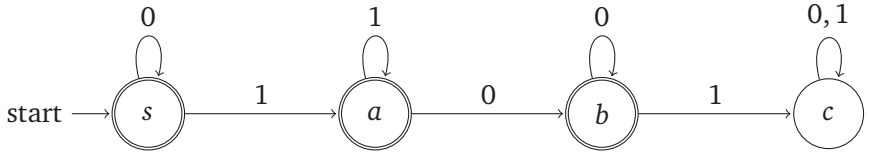
- s: strings are empty
- a_1: strings start with 0 and ends with 0
- b_1: strings start with 0 and end with 1
- a_2: strings start with 1 and end with 1
- b_2: strings start with 1 and end with 0

(b) Regular Expression: $0^*1^*((00+000)^*1^*)^*(\epsilon+0)$



- s: strings do not contain substring 101, end with 0 but not 10
- a: strings do not contain substring 101 and end with 1
- b: strings do not contain substring 101 and end with 10
- c: strings contain substring 101

(c) Regular Expression: $0^*1^*0^*$



- s: strings only contain 0
- a: strings do not contain sequence 101 and end with 1
- b: strings do not contian sequence 101 but contain 10
- c: strings contain sequence 101

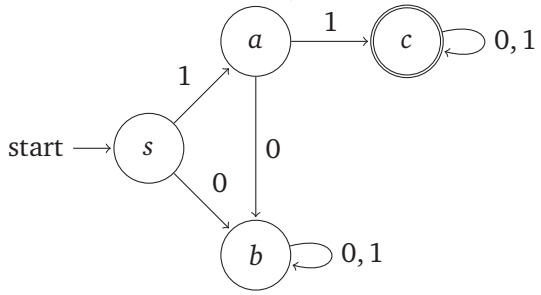
■

CS/ECE 374 B Fall 2019 Jiashuo Tong (jtong8)

Homework 1 Problem 3

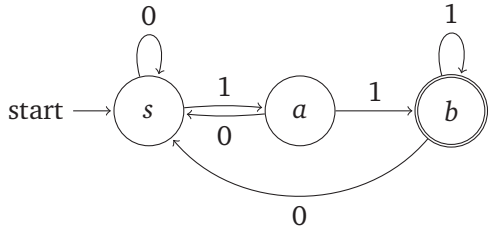
Solution:

1. $L(M_1)$ contains all strings that start with 11



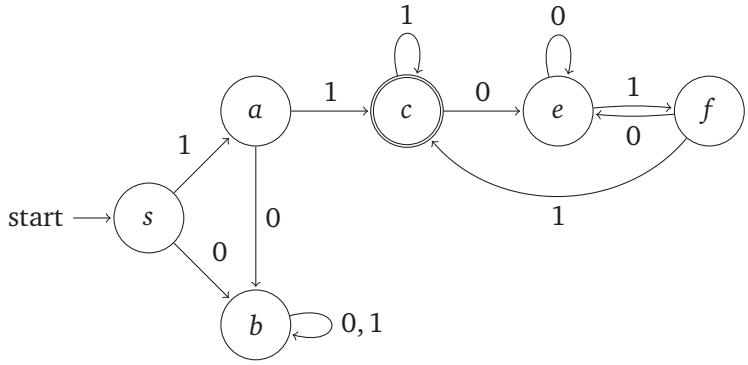
   - s: strings are empty
   - a: strings start with 1
   - b: strings start with 10
   - c: strings start with 11

2. $L(M_2)$ contains all strings that end with 11



   - s: strings end with 0
   - a: strings end with a single 1
   - b: strings end with 11

3. $L(M_3)$ contains all strings that start with 11 and end with 11



   - s: strings are empty

- a: strings start with 1
- b: strings start with 10
- c: strings start with 11 and end with 11
- e: strings start with 11 but end with 0
- f: strings start with 11 but end with 01

Why my construction satisfies each of the conditions?

- $|Q_1| = |Q_2| = 3$
- M_1 and M_2 can use fewer number of states for recognizing $L(M_1)$ and $L(M_2)$
- $L(M_1)$ start with 11 and $L(M_2)$ end with 11, so they are different languages
- $L(M_3)$ start and end with 11, so M_3 recognizes both $L(M_1)$ and $L(M_2)$
- There are infinitely many strings that start and end with 11
- $|Q_3| = 6$ which is less than $|Q_1| \times |Q_2| = 9$

■