Ozan Cinci
2448223

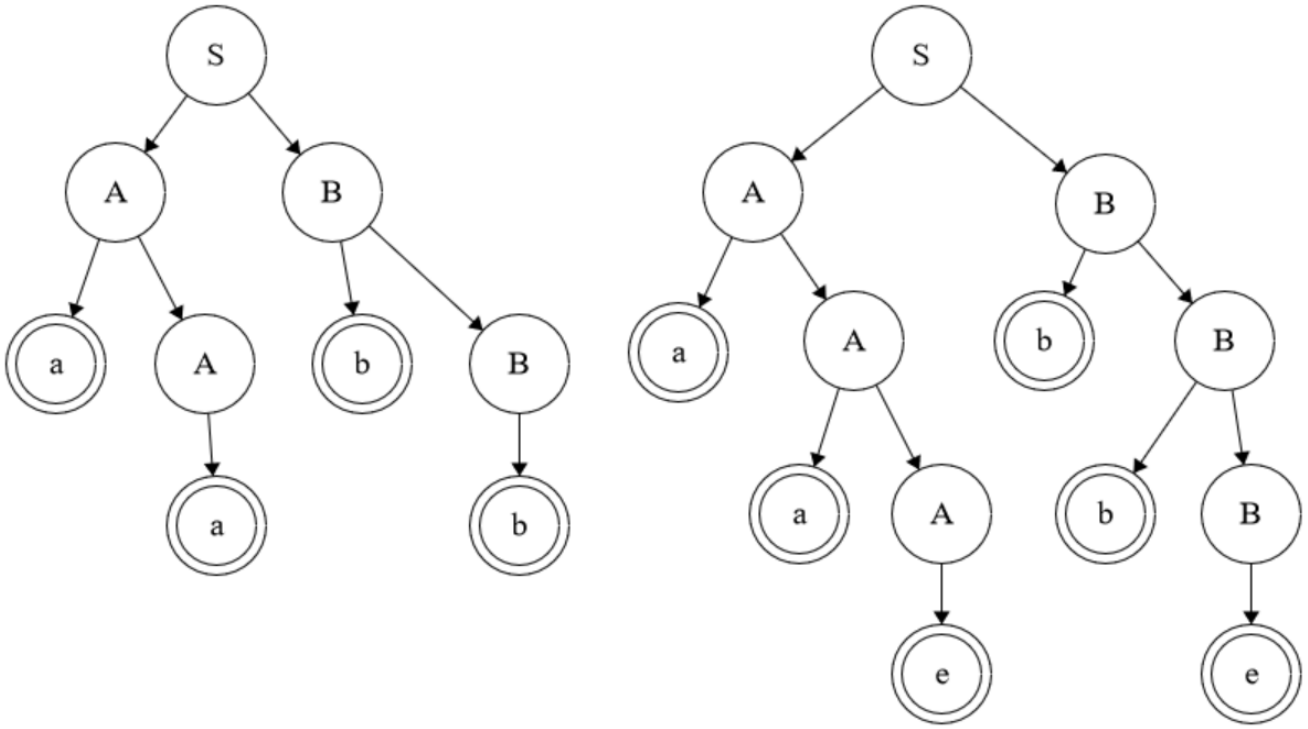
HW5

Q1.

a) It represents all the strings in form 1^n0^n or 0^n1^n
b) No. It is not ambiguous. It generates exactly one parse tree per string.

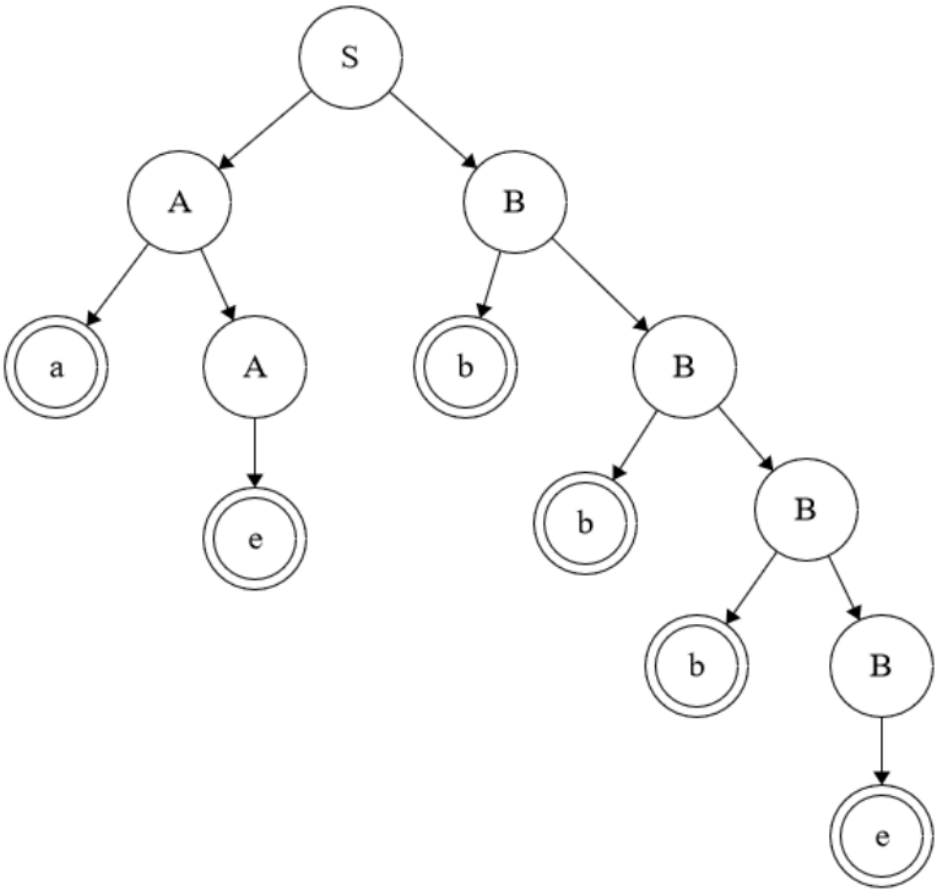
Q2.

a) Consider the string “aabb”. We can generate two different parse trees. It shows that G_2 is ambiguous.



b) G = {V, Σ, R, S} where V = {a, b, S, A, B}, Σ = {a, b} and R is;
S → AB
A → aA|ε
B → bB|ε

c) S → AB → aAB → aεB → aεbB → aεbbB → aεbbbB → aεbbbε → abbb



Q3.

a)

i) L1 is a deterministic context free language. We can create a DPDA to describe the behavior of L1. For the sake of readability, I will refer to L1.a (left set) and L1.b (right set) as the two language sets that constitute L1 by union operation. First of all, there will be a start state. In that start state, there will be two transitions which are not compatible. The first transition is made by consuming c, and the other one is by d. Both of them mark the bottom of the stack. After consuming the first letter and moving from initial to one of two separate states, the behavior of the rest of the system will be completely different. The first behavior is by consuming c. In the current state, as long as we read a, push it into the stack and stay in the same state. When counter b, pop the top most element and move to the next state. As long as we

read b, keep popping the stack. If we consume all the string we read, and the stack is not empty then move with an empty transition to one of the final states and empty the stack by popping, then it is accepted. Else if the stack is empty (the only element is the last element) but we still have the remaining string, then make a transition consuming b and popping the stack bottom marker into one of the final states. Then read input without changing the state, and the stack until all the string is accepted. These were the two conditions that L1.a accepts as a string. Now defining L1.b: after initial transition, we push “aa” into stack everytime we read a. When we read b, go to the final state and pop “a” from the stack everytime we read b. After consuming all “a” elements in the stack, read “e” by popping the stack bottom marker ”d” to move into one of the final states with empty stack, so the string is accepted. As you can see, there are no compatible transitions and ambiguity, it can be depicted by DPDA so it is a deterministic context free language.

ii) L2 is a deterministic context free language. We can create a DPDA to describe the behavior of L2. For the sake of readability, I will refer to L2.a (left set) and L2.b (right set) as the two language sets that constitute L2 by union operation. The initial state pushes “aa” everytime it consumes the letter “a”. It keeps going as long as we read “a”. When “c” or “d” is encountered, this is where separation begins. The rest of the system consists of two separate parts. Assume the first part is for L2.a and the second part for L2.b. They both read b and pop elements from stack. For L2.a, pop the top most element “aa” as long as we read b, keep popping the stack. If we consume all the string we read, and the stack is not empty then move with an empty transition to one of the final states and empty the stack by popping, then it is accepted. Else if the stack is empty (the only element is the last element) but we still have the remaining string, then make a transition consuming b and popping the stack bottom marker into one of the final states. Then read input without changing

the state, and the stack until all the string is accepted. These were the two conditions that L2.a accepts as a string. Now defining the remaining part of L2.b, when we read b, pop “a” from the stack everytime we read b. After consuming all “a” elements in the stack, read “e” by popping the stack bottom marker ”d” to move into one of the final states with empty stack, so the string is accepted. As you can see, there are no compatible transitions and ambiguity, it can be depicted by DPDA so it is a deterministic context free language.

b) Here is the graph. The blue area represents the complements of CFL. Since CFL are not closed under complementation while DCFL and RL are, the blue area corresponds to the complementable area.