Homework #4

Artemis Kelly
CS 424
Emory University

April 22, 2020

Problem 1: 2.26

Show that if G is a CFG in Chomsky normal form, then for any string w $\in$ L(G) of length n $\geq$ 1, exactly 2n - 1 steps are required for any derivation of w.
If G is a CFG in Chomsky normal form, we know that the rules in G are in the form of either A $\rightarrow$ BC or A $\rightarrow$ a. Use of the first rule results in an increase in the number of variables by one, from one variable to two variables. Use of the second rule results in a decrease in the number of variables by one and an increase of the terminals by one. In order to count how many steps are required for a derivation of w, we need to know how many times we will use the first rule, and how many times we may use the second. We can count this because we know the string must end up being n variables and that each variable must be converted into a terminal.
In order to do this lets make i = the amount of times we use the first rule and j = the amount of times we use the second rule. Then, the total number of steps required is i+j. We know that j = n because the resulting string w is length n and only the equation j can be used to convert from a variable to a terminal. Therefore it takes n steps to convert each variable into a terminal. In addition, we know that we must, in some order, grow our string from the original variable to the n variables, which can then be converted, and this will take n-1 steps. Therefore, we also know 1 + i - j = 0. The 1 represents the starting variable, i is the uses of the first transition, and j is the uses of the second transition.
If you plug in n for j in the second equation, you get i = n-1. Therefor, total steps is i + j = n-1 + n = 2n-1

Problem 2: 4.2

Consider the problem of determining whether a DFA and a regular expression are equivalent. Express this problem as a language and show that it is decidable.
We know that EQ_{DFA} is a decidable language (theorem 4.5) which shows equivalence between two DFA's. Using this previously defined decider, we can construct a language EQ_{DR} that will express the problem of determining whether a DFA and a regular expression are equivalent. We will do so by converting the Regular expression into an NFA and then converting that NFA into a DFA. Then, we will just use the TM shown after theorem 4.5 to compare the two.

EQ_{DR} = { ⟨ D, R ⟩ ‖ D is a DFA and R is a regular expression and L(D) = L(R)}
To prove that EQ_{DR} is a decidable language, I will construct a turing machine M that decides the language EQ_{DR}.
Proof: The following TM M decides EQ_{DR}
M = "On input ⟨ D, R ⟩, where D is a DFA and R is a regular expression,

1.) We follow the procedure outlined after Lemma 1.55 to convert our regular expression R into an NFA, R_N, which recognizes the same language, L(R).
2.) We follow the procedure outlined after theorem 1.39 to convert R_N into a DFA recognizing the same language. Lets call this DFA R_D.
3.) Now, using the procedure outlined in theorem 4.5, we can check if D, our original DFA and R_D, our DFA which recognizes the same language as our original regular expression, are equivalent. We run TM F from theorem 4.5 on an input of ⟨ D, R_D ⟩.
4. If F accepts, M accepts. If F rejects, reject."

Problem 3: 4.3

Let ALL_{DFA} = {⟨ A ⟩ ‖ A is a DFA and L(A) = Σ^* }. Show that ALL_{DFA} is decidable.
This proof is pretty much the opposite of the proof of E_{DFA} which proves that the DFA accepts no strings. In this case, we want to create a DFA that accepts all strings possible with the alphabet. We can use the proof in theorem 4.4 with TM T to prove this is decidable. We are able to use TM T in reverse. By switching accepting and rejecting states, if all strings are rejected, we know all strings would be accepted and can accept. To prove that decidability of ALL_{DFA}, I will construct a turing machine W which decides ALL_{DFA}.
Proof: The following TM W decides ALL_{DFA}
W = "On input ⟨ A ⟩, where A is a DFA

1.) Create a DFA X which is found by copying A and interchanging the accept and reject states.
2.) Run TM T from theorem 4.4 on input ⟨ X ⟩.
3.) If T rejects, W can reject. If T accepts, W can accept."

Problem 4: 4.7

Let B be the set of all infinite sequences over 0,1. Show that B is uncountable using a proof by diagonalization.
We will show that B is uncountable by using contradiction by diagonalization. To start, we will assume that B is countable. We are assuming that there is a correspondence f: N → B. Each element in B is an infinite sequence over {0,1}, so for (b_1, b_2,...) each $b_i \in$ {0,1}.
To define the correspondence, for each n ∈ N, we let f(n) = (b_{n1}, b_{n2},...). So b_{nk}, for example, would be the kth number in the nth sequence which lines up with n ∈ N.
Now we define an infinite sequence h = (h_1, h_2, h_3,...) ∈ B. For each $h_i \in$ {0,1}, h_i = 1 - b_{ii}. The effect of this follows the lay out of theorem 4.17. Going down sequence by sequence, the first digit in h will not match the first digit of the sequence paired up with n = 1, the

second digit will not match the second digit of the sequence paired up with n = 2, and so on. Counting on diagonally on a table for this, we obtain all the digits of h and we know that h is not f(n) for any n because it differs from f(n) in the nth digit. This is a contradiction and as such, B is uncountable.

n	f(n)
1	$(b_{11}, b_{12}, b_{13}...)$
2	$(b_{21}, b_{22}, b_{23}...)$
3	$(b_{31}, b_{32}, b_{33}...)$
4	$(b_{41}, b_{42}, b_{43}...)$

Problem 5: 5.1

Show that EQ_{CFG} is undecidable.
We will assume that EQ_{CFG} is decidable and we will use that assumption to show that ALL_{CFG} is decidable. We will do so by reducing ALL_{CFG} to EQ_{CFG}.
ALL_{CFG} = { ⟨ C ⟩ ∥ C is a CFG and L(C) = Σ^*}
EQ_{CFG} = { ⟨ G_1, G_2 ⟩ ∥ G_1 and G_2 are CFGs and L(G_1) = L(G_2)}
Assume we have a TM R that decides EQ_{CFG}. Then we will use R to construct S, a TM that decides ALL_{CFG}.
S = "On input ⟨ C ⟩, an encoding of CFG C over the set of terminals Σ:

1.) Let C_1 be a context-free grammar such that L(C_1) = Σ^*. That is to say C_1 has a start variable S and for every x ∈Σ there is a rule S → xS. It also contains the rule S → ϵ.
2.) Run R on input ⟨ C, C_1 ⟩
3.) If R accepts, S will accept. If R rejects, S will reject.

From this construction, we can see that ALL_{CFG} is reducible to EQ_{CFG}. Clearly, if R decides EQ_{CFG}, then S decides ALL_{CFG} . However, we know ALL_{CFG} is undecidable so EQ_{CFG} must also be undecidable.

Problem 6: 5.2

Show that EQ_{CFG} is co-Turing-recognizable.
A language is co-Turing recognizable when it's complement is Turing-recognizable. The complement of this language is comparing two CFGs and seeing that their languages are not the same. I will construct a TM which is a recognizer for the complement of EQ_{CFG}.
M = "On input ⟨ C_1, C_2 ⟩ where C_1 and C_2 are CFGs over the alphabet Σ:

1.) Generate all strings from Σ^* in alphabetical / lexicographical order. Lets call these strings x. (We know this is possible because theorem 4.7 says A_{CFG} is decidable)
2.) For each string x, check if x is in one language and not the other. That is, check is x ∈ L(C$_1$) and x ∉ L(C$_2$) or x ∈ L(C$_2$) and x ∉ L(C$_1$). If either of these is true, accept. Otherwise, move on to the next string for step 2.

M is a recognizer for the compliment of EQ_{CFG} because it will only accept if L(C_1) ≠ L(C$_2$). If the two languages are equal, it will never accept and it will loop, which is why it is not a recognizer for EQ_{CFG}. However, if the two languages are not equal, eventually it will find a string which is in one and not the other and accept. Therefore, EQ_{CFG} is co-turing-recognizable.

Problem 7: 5.3

Find a match in the following instance of the Post Correspondence Problem.
$\{ \frac{ab}{abab}, \frac{b}{a}, \frac{aba}{b}, \frac{aa}{a} \}$
We can find a match for the given collection of "dominos" by using the first, second and fourth domino. Putting them in the order of 4, 4, 2 and then 1, we get the same sequence on the top and bottom of "aaaabab", shown bellow.
$\{\frac{aa}{a}, \frac{aa}{a}, \frac{b}{a}, \frac{ab}{abab}\}$

Problem 8: 5.30b

Use Rice's theorem, which appears in Problem 5.28, to prove the undecidability of each of the following languages. {⟨M⟩— M is a TM and 1011 L(M)}
Allow P = {⟨M⟩— M is a TM and 1011 L(M)}. Rice's theorem requires that P be a language consisting of turing machine descriptions where P must fulfill two conditions. The first condition is P must be nontrivial - it must contain some but not all TM descriptions. The second condition is P is a property of the TMs language. What this means is
Allow P = {⟨M⟩— M is a TM and 1011 L(M)}. P is then a language of TM descriptions which satisfies the conditions of Rice's Theorem.
The first condition: "P is non trivial". P is nontrivial because some, but not all TMs contain 1011 in their language.
The second condition: P is a property of the TM's language, it depends only on the language. This means if TMs recognize the same language, then they will both have descriptions in P or neither of them will. both languages have descriptions in P because both of their languages contain 1011 or neither have descriptions in P.
Because of these two conditions, Rice's Theorem shows that P is undecidable.