COMP 330 Winter 2021
Assignment 4

Belle Pan 260839939

11th March 2021

Solution 1.
A sequence of parentheses is a sequence of (and) symbols or the empty sequence. Such a sequence is said to be balanced if it has an equal number of (and) symbols and in every prefix of the sequence the number of left parentheses is greater than or equal to the number of right parentheses. Thus (((())())) is balanced but, for example, ())(is not balanced even though there are an equal number of left and right parentheses. The empty sequence is considered to be balanced. Consider the grammar $S \to (S)|SS|\epsilon$. This grammar is claimed to generate balanced parentheses.

1. **Prove that any string generated by this grammar is a properly balanced sequence of parentheses.**
2. **Prove that every sequence of balanced parentheses can be generated by this grammar.**

.

Part 1: we prove this using induction on products of the grammar.

- Base case: ϵ.
  - This is clearly balanced, as stated in the problem.
- Inductive hypothesis: All words up to length n generated by the grammar are properly balanced.
  - Consider a word w' of length $n+1$:
    - Case 1: $S \to (S)$
      - w' is a word of the form (w) where w is a word generated by the grammar and $|w| = n$.
      - By the inductive hypothesis, w is properly balanced and so should w'.
    - Case 2: $S \to SS$
      - w' is a word of the form w_1w_2 where w_1 and w_2 are both generated by the grammar and $|w_1|$ and $|w_2| \leq n$.
      - By the inductive hypothesis, w_1 and w_2 are both properly balanced, and so should w'.
- The inductive hypothesis holds and thus shows that any word generated by the grammar is a properly balanced sequence of parentheses.

Part 2: we prove this using induction on the length of words in the language.

- Base case: ϵ.
  - This is clearly balanced, and can be generated by the grammar through $S \to \epsilon$.
- Inductive hypothesis: All words up to length n in the language can be generated by the grammar.
  - Consider a word w', where $|w'| = n+1$ and w' is in the language (i.e. properly balanced). We will utilize a stack to ensure that the parentheses are balanced. If, when reading a word, a (is seen, it is pushed onto the stack; then, when a) is read, we pop the first element off the stack. If the word is balanced, the stack will be empty at the end of the word.
    - Case 1: the stack is empty before the end of w' and empty again at the end of w'.
      - We know that w' must be in the form w_1w_2, where the stack is empty at the end of reading w_1 and at the end of reading w_2. This indicates that both w_1 and w_2 are balanced.

- We can see that a word of this form can be generated by the grammar using $S \to SS$. Because $|w_1|$ and $|w_2| \leq n$, they can be generated by the grammar according to the inductive hypothesis; thus, w' can also be generated by the grammar.

* Case 2: the stack is empty at the end of w'.
  - We know that w' cannot be split into two or more words that are balanced like in Case 1; however, we can rewrite w' as (w) where w has length $n-1$. This indicates that w must be balanced.
  - Words of this form can be generated by the grammar using $S \to (S)$. By the inductive hypothesis, w can be generated by the grammar as $|w| \leq n$; thus, w' can also be generated by the grammar.

- The inductive hypothesis holds and thus shows that every sequence of balanced parentheses can be generated by this grammar.

Solution 2.
Consider the following context-free grammar : $S \to aS|aSbS|\epsilon$. This grammar generates all strings were in every prefix the number of a's is greater than or equal to the number of b's. Show that this grammar is ambiguous by giving an example of a string and showing that it has two different derivations.

Below are two ways to generate the string aab using the grammar :

- $S \to aS \to aaSbS \to aabS \to aab$
- $S \to aSbS \to aaSbS \to aabS \to aab$

As there are two different ways to generate the same word using this grammar, it is clear that this grammar is ambiguous!

Solution 3.
We define the language $PAREN_2$ inductively as follows:

1. $\epsilon \in PAREN_2$,
2. **if** $x \in PAREN_2$ **then so are** (x) **and** $[x]$,
3. **if** x **and** y **are both in** $PAREN_2$ **then so is** xy.

Describe a PDA for this language which accepts by empty stack. Give all the transitions.

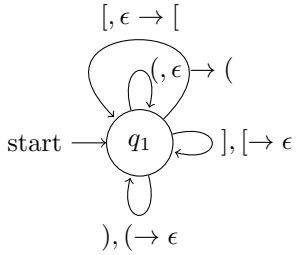

Figure 1: A PDA for $PAREN_2$ that accepts by empty stacks.

(and [are pushed onto the stack and are popped off when a matching parenthesis is read; (matches only with) and [matches only with]. When the machine encounters non-matching parentheses - when a closing parenthesis read but it is not a match for the one at the top of the stack - , it jams and rejects. The machine accepts only at the end of the word if the stack is empty - i.e. when all parenthesis are matched properly. This machine accepts by empty stack, so there is no need to have accept states.

Solution 4.
Consider the language $\{a^n b^m c^p | n \leq p$ or $m \leq p\}$. Show that this is context-free by giving a grammar. You need not give a formal proof that your grammar is correct but you must explain, at least briefly, why it works.

The following is the grammar:

$S \to NC|AMC$
$N \to aNc|B$
$M \to bMc|\epsilon$
$A \to aA|\epsilon$
$B \to bB|\epsilon$
$C \to cC|\epsilon$

Strings generated by this grammar are all in the language:

- S gives us two choices:
  - NC where the N ensures that the number of a's $\leq$ the number of c's in the word and allows any number of b's in the word ($n \leq p$).
  - AMC where the M ensures that the number of b's $\leq$ the number of c's in the word and the A allows any number of a's in the word ($m \leq p$).
  - C at the end of each option allows us to add any number of c's to the end of the word.
- Thus, all words generated by this grammar are in the language.

All strings in the language can be generated by the grammar:

- Given an arbitrary word $w = a^i b^j c^k$, we can generate w as follows:
  - First, determine which of i and j is smaller - we denote this value l and rewrite w as $a^i b^j c^l c^{k-l}$.
  - Case 1: $i < j$ such that $a^i b^j c^l c^{k-l} = a^i b^j c^i c^{k-i}$
    - We use $S \to NC \to aNcC$ to generate this word.
    - To ensure that the number of a's is equal to the number of b's, we simply need to repeat the instruction $N \to aNc$ i times. On the $i+1$ time we execute N, we use the instruction $N \to B \to bB$ and repeat $B \to bB$ j times to generate the number of b's in the word and terminating the b's using $B \to \epsilon$ on the $j+1$ iteration.
    - We generate c^{k-i} by repeating $C \to cC$ $k-i$ times and then terminating using ϵ on the $k-i+1$ iteration.
    - This word terminates because B and C both have the option of ϵ.
  - Case 2: $j < i$ such that $a^i b^j c^l c^{k-l} = a^i b^j c^j c^{k-j}$
    - We use $S \to AMC$ to generate this word.
    - A generates the number of a's in the word
    - $M \to bMc$ ensures that the number of b's generated equals the number of c's generated in the word. We simply need to repeat the $M \to bMc$ instruction j times to generate this portion of the word and then terminating the number of b's added on the $j+1$ time using $M \to \epsilon$.
    - We generate c^{k-i} by repeating $C \to cC$ $k-j$ times and then terminating using ϵ on the $k-j+1$ iteration.
    - This word terminates because A, M and C have the option of ϵ.
- Thus, all words in the language may be generated using this grammar.

Solution 5.
A linear grammar is one in which every rule has exactly one terminal and one non-terminal on the right hand side or just a single terminal on the right hand side or the empty stack. Here is an example : $S \to aS|a|bB$ **;** $B \to bB|b$**.**

1. **Prove that any regular language can be generated by a linear grammar.**
2. **Is every language generated by a linear grammar regular? If your answer is "yes" you must give a proof. If you answer is "no" you must give an example.**

.
Part 1 : prove any regular language can be generated by a linear grammar.

- Let there be a language L with a fixed alphabet Σ that has a DFA, $M = (S, s_0, \delta, F)$.
- Let the start symbol (S) of the grammar be denoted by the start state of M.
- Let the non-accept states of M represent the non-terminal rules in the grammar. For each non-accept state x with transition $\delta(x, y) = z$, we have a non-terminal rule in the form $X \to yZ$.

- Let the accept states of M represent the terminal rules in the grammar. For each accept state x, we have a terminal rule in the form of $X \to \epsilon$.
- Clearly, M can be created on the basis of these rules.
- When a string is passed through M, we can easily get the sequence of rules we need to follow to produce the string.

Part 2 : NOT every language generated by a linear grammar is regular.

- Take, for example, the grammar:
  - $S \to aSb|\epsilon$
- This is clearly a linear grammar with only one terminal and one non-terminal on the right hand side of S, yet it generates the language $\{a^n b^n | n \geq 0\}$ which we know is a non-regular language.
- Thus, not every language generated by a linear grammar is regular.