CPSC 313 — Winter 2020
Assignment 3 — Non-Context-Free Grammars and Turing Machines

Haohu Shen

30063099

1. **Pumping Lemma for context-free languages**

   Problem 2.31, page 157 (3rd ed) and page 131 (2nd ed) of Sipser. For convenience, the question is reproduced here. Recall that a *palindrome* is a string that reads the same forwards as backwards. Let L be the language of all palindromes over the alphabet $\{0, 1\}$ that contain an equal number of 0's and 1's. For example, 0110 and 010110011010 belong to L, but 0101 and 010 do not. Use the Pumping Lemma for context-free languages to prove that L is not context-free.

   Solution. We prove that L is not context-free by contradiction. We assume that L is context-free, then it satisfies the Pumping Lemma for context-free languages, thus we can let p be the pumping length of L such that $s = 0^p1^{2p}0^p$, thus s is a palindrome string such that $s \in L$ and $|s| = p+2p+p = 4p \geq p$. Therefore, according to the Pumping Lemma, s can be devided into 5 pieces of strings $u, v, x, y, z \in \sum^*$ such that $s = uvxyz$ satisfying the following statements

   - $uv^ixy^iz \in L$ for every integers $i \geq 0$
   - $|vy| > 0$
   - $|vxy| \leq p$

   According the first statement, we let $i = 0$ and we have $uv^ixy^iz = uv^0xy^0z = uxz \in L$. Since $|vy| > 0$, we have $v \neq \varepsilon$ or $y \neq \varepsilon$. Since $|vxy| \leq p$, it is impossible that vxy contains 0s from both the leftmost 0^p of s and the rightmost 0^p of s since the length of the substring between them is $2p$ and $2p > p$, thus we can discuss the possible forms of the substring vxy in 2 cases:

   Case 1: If v does not contain any 0s and y does not contain any 0s, that is, the substring 1^{2p} of s contains the whole vxy, since $v \neq \varepsilon$ or $y \neq \varepsilon$, v or y contains at least one 1. Thus, when we obtain uxz by removing v and y from s, the number of the symbol 1 in s will be also decreased, therefore, if we suppose that $uxz = 0^p1^m0^p$ we will have $m < 2p$, which contradicts that $uxz \in L$.

   Case 2: If v or y contains a 0, it indicates that vxy contains 0s only from the leftmost 0^p of s or only from the rightmost 0^p of s, thus we can split the case into 2 subcases.

   - If vxy contains 0s only from the leftmost 0^p of s, when we obtain uxz by removing v and y from s, the number of the symbol 0 in the leftmost 0^p will be decreased, since the number of the symbol 1 in the middle substring 1^{2p} of s may or may not decrease, we can suppose that uxz has the form $uxz = 0^n1^m0^p$ such that $n < p$ and $m \leq 2p$, which contradicts that $uxz \in L$.

- If vxy contains 0s only from the rightmost 0^p of s, when we obtain uxz by removing v and y from s, the number of the symbol 0 in the rightmost 0^p will be decreased, since the number of the symbol 1 in the middle substring 1^{2p} of s may or may not decrease, we can suppose that uxz has the form $uxz = 0^p1^m0^n$ such that $n < p$ and $m \leq 2p$, which contradicts that $uxz \in L$.

Therefore, we can conclude that, L is not context-free.

2. **Closure properties of Turing recognizable languages**

Prove that the set of Turing recognizable languages is closed under the operations of concatenation and Kleene closure.

Proceed as follows. Let L_1 and L_2 be Turing-recognizable languages over some alphabet Σ, and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be Turing machines that recognize L_1 and L_2, respectively, so $L(\mathcal{M}_1) = L_1$ and $L(\mathcal{M}_2) = L_2$. Construct Turing machines $\mathcal{M}$ and $\mathcal{M}^*$ that recognize L_1L_2 and L_1^*, respectively. It is easiest to construct these as a *non-deterministic* Turing machine, so you can non-deterministically split up your input string as a concatenation (of two strings, i.e. $w = w_1w_2$, for the concatenation and of an arbitrary number of strings, i.e. $w = w_1w_2 \dots w_n$, for the Kleene closure).

For both operations, prove your construction correct. That is, prove that $L(\mathcal{M}) = L_1L_2$, $L(\mathcal{M}^*) = L_1^*$, and acceptance for both $\mathcal{M}$ and $\mathcal{M}^*$ happens in a finite number of steps.

Solution. We prove the statement by the construction of a non-deterministic Turing machine $\mathcal{M}$ such that it is able to recognize L_1L_2 and L_1^*, the Turing machine $\mathcal{M}$ will take finite number of steps for acceptance or rejection since the number of splits is finite for the Turing machine $\mathcal{M}_1$ and the Turing machine $\mathcal{M}_2$.

Case 1: We firstly prove that the set of Turing recognizable languages is closed under the operations of concatenation. We suppose $\mathcal{M}$ has 3 tapes and follow the steps below:

- First of all, since the input string w will be contained in the 1st tape, we split w into two substrings w_1 and w_2 non-deterministically such that $w = w_1w_2$.
- We copy w_1 to the 2nd tape.
- We copy w_2 to the 3rd tape.
- We run $\mathcal{M}_1$ in the 2nd tape to process w_1, and if $\mathcal{M}_1$ rejects, $\mathcal{M}$ rejects.
- We run $\mathcal{M}_2$ in the 3rd tape to process w_2, and if $\mathcal{M}_2$ rejects, $\mathcal{M}$ rejects as well.
- Therefore, it must be the case that $\mathcal{M}$ accepts w if both $\mathcal{M}_1$ accepts w_1 and $\mathcal{M}_2$ accepts w_2. Thus $\mathcal{M}$ is a non-deterministic decider and we convert it to a Turing machine that has a single tape. And now a Turing machine for the operations of concatenation is constructed.

Case 2: Now we prove that the set of Turing recognizable languages is closed under the operations of Kleene closure. We suppose $\mathcal{M}$ has 2 tapes and follow the steps below:

- First of all, we choose the leftmost non-empty part of the input w that has not been read yet, and then we copy and paste it to the 2nd tape, that is, we split w into n parts such that $w = w_1w_2 \dots w_n$ and what we select is w_1.
- And then we run $\mathcal{M}_1$ on the 2nd tape to process the part we selected in the first step and we will have 3 subcases:
  - If $\mathcal{M}_1$ rejects, then $\mathcal{M}$ rejects.
  - If $\mathcal{M}_1$ accepts and w is fully processed, then $\mathcal{M}$ accepts.
  - If $\mathcal{M}_1$ accepts but w is not fully processed, we clean the 2nd tape and return to the 1st step and select the next leftmost part of w.

- Since $\mathcal{M}$ is non-deterministic, w_k is being split non-deterministically such that $1 \leq k \leq n$, we check whether every w_k belongs to the L_1 and L_1^* or not. Furthermore, if $|w|$ is finite, there must be a finite number of string split as well, thus $\mathcal{M}$ will take a finite number of steps to decide acceptance or rejection.
- Finally, we convert $\mathcal{M}$ to a Turing machine that has a single tape, now a Turing machine for the operations of Kleene closure is constructed.

Therefore, we can conclude that, the set of Turing recognizable languages is closed under the operations of concatenation and Kleene closure.

3. **Turing machine design**

Consider the language $L_{n\text{-steps}}$ consisting of all encodings $\langle \mathcal{M}, w, n \rangle$ where

- $\mathcal{M}$ is Turing machine,
- w is a string,
- n is a positive integer and
- $\mathcal{M}$ halts on input w in exactly n steps (i.e. in n steps but n no fewer that n steps).

Prove that $L_{n\text{-steps}}$ is decidable by constructing a decider $\mathcal{M}_{n\text{-steps}}$ for $L_{n\text{-steps}}$. Prove your decider correct.

Solution Let $L_{n-steps} = \{\langle \mathcal{M}, w, n \rangle | \mathcal{M} \text{ accepts } w \text{ in exactly } n \text{ steps}\}$, we construct a $\mathcal{M}_{n\text{-steps}}$ for $L_{n\text{-steps}}$ in order to prove $L_{n\text{-steps}}$ is decidable as follow:

- We firstly filter the input by validate whether the input has the correct form or not, reject if its form is incorrect.
- Then we suppose $\mathcal{M} = \mathcal{M}_{n\text{-steps}}$ has 4 tapes and
  - we copy the start state of $\mathcal{M}$ to the 2nd tape.
  - we copy w to the 3rd tape.
  - we make the 4th tape to count the steps for us, in order to do, we write n 0s on it.
- Now we simulate the Turing machine $\mathcal{M}$ by the steps below:
  - From the 2nd tape, we read the current state of $\mathcal{M}$ and name it m.
  - From the 3rd tape, we read the current symbol being processed of the input w and name this symbol c.
  - From the 1st tape, we search for the transition $\delta(m, c)$ and call this transition, and:
    - reject if any reject state of $\mathcal{M}$ is encountered.
    - accept if any accept state of $\mathcal{M}$ is encountered.
    - loop if no halt state of $\mathcal{M}$ is encountered.
  - If a move is executed, remove a zero on the 4th tape, and if the 4th tape is blank, then:
    - reject if any reject state of $\mathcal{M}$ is encountered.
    - reject if no halt state of $\mathcal{M}$ is encountered.
    - accept if any accept state of $\mathcal{M}$ is encountered.
  - Reject if halt state of $\mathcal{M}$ is encountered before the 4th tape is blank.
  - Reject if halt state of $\mathcal{M}$ is encountered after the 4th tape is blank.
- Therefore $\mathcal{M}$ can decide the acceptance or rejection of the input w in exactly n steps.
- Since such decider is constructed, we can conclude that $L_{n\text{-steps}}$ is decidable.