# CPSC 313 — Winter 2020
# Assignment 2 — Context-Free Languages and Grammars

Haohu Shen

30063099

1. **Non-regular languages and the Pumping Lemma**

   Let $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, =\}$ and consider the language $L$ of all strings over $\Sigma$ that constitute a valid and correct equation of the form $a + b = c$ where $a, b, c$ are non-negative integers represented in base 10, without leading zeros. Some elements of $L$ include $13+17 = 30$ and $99 + 0 = 99$, but not $13 + 17 = 29$ (wrong arithmetic) or $99 + 01 = 100$ (leading zero in the number 1). Use the Pumping Lemma to prove that $L$ is not regular.

   **Solution.** We prove $L$ is regular by contradiction. Assume that $L$ is regular. Then $L$ satisfies the Pumping Lemma. Let $p$ be the pumping length of $L$ and consider the string $s$ such that it denotes $1^p + 0 = 1^p$ where $1^p$ is an integer of size $p$ that contains 1 only, thus we have $s \in L$. Since $|s| = p + 3 + p = 2p + 3 \geq p$, by the Pumping Lemma, $s$ can be written as $s = xyz$ with strings $x, y, z \in \Sigma^*$ such that

   - $y \neq \varepsilon$
   - $|xy| \leq p$
   - $xy^i z \in L$ for all $i \geq 0$

   Since $|xy| \leq p$, $xy$ only contains 1's, thus $y$ only contains 1's. Since $y \neq \varepsilon$, $y$ contains at least one 1. Thus $y$ can be write as $1^i$ such that $i \geq 1$. Consider the string $w = xy^2$, we have $w = xy^2 = xyyz$ denotes the string $1^{p+i} + 0 = 1^p$, thus $w \notin L$ since $1^{p+i} \neq 1^p$, which contradicts that $w \in L$ from the third property of the Pumping Lemma. Thus, the assumption is wrong and $L$ is not regular.

2. **Regular languages are context-free**

   Formally prove that every regular language is context-free. Use the following ingredients in your proof.

   - The recursive definition of regular expressions;
   - The fact that $L$ is a regular language if and only if $L = L(e)$ for some regular expression $e$;
   - Strong induction on the length of a regular expression — recall that every regular expression is a string of length at least 1 consisting of elements in $\Sigma$ as well as the symbols $\cup, *, (, ), \varepsilon, \emptyset$;
   - The fact that context-free languages are closed under the regular operations; that is, if $L_1$ and $L_2$ are any context-free languages, then $L_1 \cup L_2$, $L_1 L_2$ and $L_1^*$ are context-free. (You may use this result without proof; we will prove it in

   **Solution.** Suppose $L$ is a language over some alphabet $\Sigma$, suppose $e$ is a regular expression such that $L = L(e)$, thus $L$ is a regular language. We prove that $L$ is context-free by using strong induction on the length of $e$, that is, $|e|$. And the case that $|e| = 1$ will be considered in the basis.

   *Basis:* When $|e| = 1$, $e$ can be one of three subcases below:

   - $e = w$ such that $w \in \Sigma$
   - $e = \varepsilon$
   - $e = \emptyset$

   Thus we can define a grammar $G = (V, \Sigma, R, S)$ such that

   - If $e = w$ such that $w \in \Sigma$, $R = \{S \to w\}$
   - If $e = \varepsilon$, $R = \{S \to \varepsilon\}$
   - If $e = \emptyset$, $R = \emptyset$

   We prove that $L = L(G)$ for all 3 subcases:

   - When $e = w$ such that $w \in \Sigma$, $L(w) = w$, since $R = \{S \to w\}$ in this case and it only contains 1 rule, the only string we can generate by $G$ is $w$, thus we have $L(G) = L(e) = L(w) = \{w\}$
   - When $e = \varepsilon$, since $R = \{S \to \varepsilon\}$ in this case and it only contains 1 rule, the only string we can generate by $G$ is $\varepsilon$, thus we have $L(G) = L(e) = L(\varepsilon) = \varepsilon$
   - When $e = \emptyset$, since $R = \emptyset$, we cannot generate any string since there is no rule in $G$, thus the language is empty, thus $L(G) = L(e) = \emptyset$.

   Therefore, we have $L = L(G)$ for all 3 subcases, thus $L$ is context free when $|e| = 1$, as required.

   **Inductive Step:** Let $k \geq 1$ be an integer. It is necessary and sufficient to use:

   *Inductive Hypothesis:* Suppose $n$ is a non-negative integer. Suppose for all $n$ such that $1 \leq n \leq k$, for any language $L = L(e)$ where $e$ is a regular expression such that $|e| = n$, $L$ is context-free.

to prove:

*Induction claim:* For any language $L = L(e)$ where $e$ is a regular expression such that $|e| = k+1$, $L$ is context-free.

Let language $L = L(e)$ where $e$ is a regular expression such that $|e| = k+1$, since $k \geq 1$, $|e| = k+1 \geq 2$, therefore, according to the recursive definition of a regular expression, the form of $e$ can be split into 3 cases:

*Case 1:* When $e$ is the union of two regular expressions, that is, $e = e_0 \cup e_1$ such that $e_0$ and $e_1$ are two regular expressions, so

$$|e| = |e_0 \cup e_1| = |e_0| + 1 + |e_1| = |e_0| + |e_1| + 1 = k+1$$

thus $|e_0| + |e_1| = k \leq k$, from the ingredients we know that $|e_0| \geq 1$ and $|e_1| \geq 1$, thus we have $|e_0| \leq k$ and $|e_1| \leq k$, thus we have that $L(e_0)$ and $L(e_1)$ are context-free by Inductive Hypothesis. From the ingredients we also know that context-free languages are closed under unions, thus we can conclude that $L = L(e) = L(e_0) \cup L(e_1)$ is context-free.

*Case 2:* When $e$ is the concatenation of two regular expressions, that is, $e = e_0 e_1$ such that $e_0$ and $e_1$ are two regular expressions, we have

$$|e| = |e_0 e_1| = |e_0| + |e_1| = k+1$$

thus $|e_0| + |e_1| - 1 = k + 1 - 1 = k$, since we have $|e_0| \geq 1$ and $|e_1| \geq 1$ from the ingredients,

$$k = |e_0| + |e_1| - 1 = |e_0| + (|e_1| - 1) \geq |e_0|$$

thus $|e_0| \leq k$, also

$$k = |e_1| + |e_0| - 1 = |e_1| + (|e_0| - 1) \geq |e_1|$$

thus $|e_1| \leq k$, thus by Inductive Hypothesis, we have that $L(e_0)$ and $L(e_1)$ are context-free. From the ingredients we know that context-free languages are closed under concatenation, thus we can conclude that $L = L(e) = L(e_0)L(e_1)$ is context-free.

*Case 3:* When $e$ is the Kleene closure of a regular expression, that is, $e = e_0^*$ such that $e_0$ is a regular expression, we have

$$|e| = |e_0^*| = |e_0^+ \cup \varepsilon| = |e_0^+| + 2 = k+1$$

thus $|e_0^+| = k + 1 - 2 = k - 1 \leq k$, thus $|e_0| \leq |e_0^+| \leq k$, thus by Inductive Hypothesis, we have that $L(e_0)$ is context-free. From the ingredients we know that context-free languages are closed under Kleene closure, thus we can conclude that $L = L(e) = L(e_0)^*$ is context-free.

**Conclusion:** Therefore, by strong induction, we can conclude that every regular language is context-free.

3. **Designing context-free grammars and languages**

(a) Design a context-free grammar for the language

$$L = \left\{a^{2i}b^j vc^j (ac)^i \mid i, j \geq 0, v \in \{a,b\}^*\right\}$$

over the alphabet $\Sigma = \{a, b, c, \}$. Your grammar must have at most 3 variables and at most 7 rules. Clearly state the variables, the terminals, the rules, and the start variable for your grammar. You need *not* formally prove your grammar correct, but you should give a concise and convincing explanation of its correctness (in case of errors, such an explanation may also secure you partial credit).

**Solution.** We design a context-free grammar $G = (V, \Sigma, R, S)$ such that $V = \{S, A, B\}$ and $R$ is consist of 7 rules:

$$\begin{aligned}
&S \to aaSac \\
&S \to A \\
&A \to bAc \\
&A \to B \\
&B \to Ba \\
&B \to Bb \\
&B \to \varepsilon
\end{aligned}$$

Firstly, we explain that for any $w \in \Sigma^*$, if $S \overset{*}{\Rightarrow} w$, then $w \in L$.

- Firstly, We start any derivation $S \overset{*}{\Rightarrow} w$ by applying the rule $S \to aaSac$ for 0 or more times, and we have the string $(aa)^i S(ac)^i$ for some $i \geq 0$, that is, $w = a^{2i}S(ac)^i$ such that $i \geq 0$.
- Secondly, we apply the rule $S \to A$ on the previous string form, and we have $w = a^{2i}A(ac)^i$.
- Thirdly, we apply the rule $A \to bAc$ on the previous string form for 0 or more times, and we have $w = a^{2i}b^j Ac^j (ac)^i$ for some $j \geq 0$.
- Next we apply the rule $A \to B$ and replace the previous string form as $w = a^{2i}b^j Bc^j (ac)^i$.
- And then we apply the rule $B \to Ba$ or the rule $B \to Bb$ by replacing $B$ with the string $Bv$ for $v \in \{a, b\}^*$, and we have $w = a^{2i}b^j Bvc^j (ac)^i$.
- Finally we apply the rule $B \to \varepsilon$ to eliminate all occurrences of $B$ from the previous string form, and we have $w = a^{2i}b^j vc^j (ac)^i \in L$ such that $i \geq 0, j \geq 0, v \in \{a, b\}^*$

Secondly, we explain that for any $w \in \Sigma^*$, if $w \in L$, then $S \overset{*}{\Rightarrow} w$. Consider a string $w \in L$, thus $w = a^{2i}b^j vc^j (ac)^i \in L$ such that $i \geq 0, j \geq 0, v \in \{a, b\}^*$, we can obtain $w$ from $S$ by

- Firstly applying $i$ times of the rule $S \to aaSac$
- Secondly applying the rule $S \to A$
- Thirdly applying $j$ times of the rule $A \to bAc$
- Next applying the rule $A \to B$

- And then apply 0 or more times of the rule $B \to Ba$ or the rule $B \to Bb$ to become $Bv$ since $v \in \{a, b\}^*$
- Finally we eliminate all occurrences of $B$ by applying the rule $B \to \varepsilon$, and we have $w$.

(b) Consider the context-free grammar $G = (V, \Sigma, R, S)$ where $\Sigma = \{a, b, c\}$, $V = \{S, A, B, C\}$, $S$ is the start variable and $R$ consists of the rules

$$
\begin{aligned}
S &\to ASA \mid B \\
A &\to a \mid b \\
B &\to BC \mid \varepsilon \\
C &\to cc
\end{aligned}
$$

Give a formal description of $L(G)$, in the form $L(G) = \{\cdots \mid \cdots\}$. You need not formally prove your language correct, but you should again give a concise, coherent, convincing explanation of how you obtained your answer. (Again, in case of errors, such an explanation may help you gain partial credit for this problem).

**Solution.** Since we notice that the rules of $G$ with variables $A$ or $C$ contain terminals only on the right, we merge these 2 rules and simplify $R$ such that the rules become:

$$
\begin{aligned}
S &\to aSa \mid aSb \mid bSa \mid bSb \mid B \\
B &\to Bcc \mid \varepsilon
\end{aligned}
$$

And we rewrite it into 4 rules in order to make our later explanation more clearly:

$$
\begin{aligned}
S &\to aSa \mid aSb \mid bSa \mid bSb \\
S &\to B \\
B &\to Bcc \\
B &\to \varepsilon
\end{aligned}
$$

We introduce $f \in \{a, b\}$ and let

$$L = \{f^j c^{2i} f^j \mid i \geq 0, j \geq 0, f \in \{a, b\}\}$$

and we explain that $L = L(G)$ in 2 steps.

Firstly we explain that for any $w \in \Sigma^*$, if $w \in L$, then $S \overset{*}{\Rightarrow} w$.

Let $w \in L$, thus $\exists i \geq 0, j \geq 0$ such that $w = f^j c^{2i} f^j$ and $f \in \{a, b\}$. We apply a finite sequence of rules of $G$, starting from the rule $S \to aSa \mid aSb \mid bSa \mid bSb$ to generate $w$:

- Firstly we apply the rule $S \to aSa \mid aSb \mid bSa \mid bSb$ for $j$ times, and we have the string form $f^j S f^j$ such that $f \in \{a, b\}$.
- Secondly we apply the rule $S \to B$ and replace the previous string form to $f^j B f^j$.
- Thirdly we apply the rule $B \to Bcc$ for $i$ times on the previous string form, and we have $f^j B (cc)^i f^j = v^j B c^{2i} f^j$.
- Finally we apply the rule $B \to \varepsilon$ to eliminate $B$ on the previous string form, which yields the string $f^j c^{2i} v^f = w$.

Secondly we explain that for any $w \in \Sigma^*$, if $S \overset{*}{\Rightarrow} w$, $w \in L$.

Let $w \in \Sigma^*$ with $S \overset{*}{\Rightarrow} w$. According to the value of $w$, we can split it into 2 cases.

*Case 1:* If $w = \varepsilon$, then $w = f^0 c^0 f^0 = f^0 c^{2 \times 0} f^0 \in L$.

*Case 2:* If $w \neq \varepsilon$, by the definition 2 on the handout *CPSC 313 — Winter 2020 Designing Context-Free Grammars*, $\exists x \in (V \cup \Sigma)^*$ such that $S \overset{*}{\Rightarrow} x \Rightarrow w$. By the definition 1 from the handout, we have

- $x = uAv, w = uzv$ where $u, v, z \in (V \cup \Sigma)^*$
- $A \in V$
- $A \to z$ is a rule in $G$.

Since $w$ consists of terminals only, $w \in L(G) \subseteq \Sigma^*$, $u, v, z \in \Sigma^*$. Since the only rule whose right-hand side does not include a variable is the rule $B \to \varepsilon$, so it must map to the rule $A \to z$ in the definition 1, thus we can obtain other mapping relations as:

- $A = B$
- $z = \varepsilon$

Thus $x = uBv$ and $w = u\varepsilon v = uv$. And the derivation $S \overset{*}{\Rightarrow} x \Rightarrow w$ becomes

$$S \overset{*}{\Rightarrow} uBv \Rightarrow uv = w$$

Since the derivation $S \overset{*}{\Rightarrow} uBv$ must contain at least one application of the rule $S \to B$ before applying the rule $B \to \varepsilon$, we consider that the first application of this rule $S \to B$ in our derivation, thus our derivation now has the form

$$S \overset{*}{\Rightarrow} t \Rightarrow y \overset{*}{\Rightarrow} x = uBv \Rightarrow uv = w$$

such that $t, y \in (V \cup \Sigma)^*$ and the derivation $t \Rightarrow y$ is the first application of the rule $S \to B$, the derivation $S \overset{*}{\Rightarrow} t$ is obtained by applying 0 or more times of the rule $S \to aSa \mid aSb \mid bSa \mid bSb$, let $m$ be the number of applications of this rule, we have $t = f^m S f^m$ such that $f \in \{a, b\}$, thus after applying the rule $S \to B$, our derivation becomes

$$S \overset{*}{\Rightarrow} t = f^m S f^m \Rightarrow y = f^m B f^m \overset{*}{\Rightarrow} x = uBv \Rightarrow uv = w$$

For the derivation $y = f^m B f^m \overset{*}{\Rightarrow} uBv$, we notice that every application of a rule whose left-hand side is $B$ to a string that contains $B$ does not increase the number of $B$ in the string and also does not introduce variables other than $B$ into the string. Also, applying the rule $B \to Bcc$ yields a string containing exactly one $B$, while applying the rule $B \to \varepsilon$ can eliminate this occurrence of $B$. Since $y = f^m B f^m$ and $x = uBv$ both contain one $B$, the derivation of $f^m B f^m \overset{*}{\Rightarrow} uBv$ consists only applications of the rule $B \to Bcc$. Thus, let a non-negative integer $n$ be the number of times the rule $B \to Bcc$ is applied, we can have

$$uBv = f^m B(cc)^n f^m = f^m B c^{2n} f^m$$

thus $u = f^m$ and $v = c^{2n} f^m$, and

$$w = uv = f^m c^{2n} f^m \in L$$