Theory of Machines and Languages - Assignment 3

Ali Abbasi - 98105879

January 12, 2023

Contents

1 Context Free Grammars

1.1

1.1.1

A grammar describing the language is as follows:

$$\begin{aligned} S &\to bS \mid aA \\ A &\to bA \mid aB \\ B &\to bB \mid aC \\ C &\to bC \mid aC \mid \epsilon \end{aligned}$$

1.1.2

$$\begin{aligned} S &\to ASA \mid 0 \\ A &\to 0 \mid 1 \end{aligned}$$

1.2

We can rewrite the language as follows:

$$L = \left\{a^i b^j c^k \mid i,j,k \geq 0, i = j\right\} \cup \left\{a^i b^j c^k \mid i,j,k \geq 0, j = k\right\} \tag{1.2.1}$$

So we have:

$$\begin{aligned} S &\to S_1 \mid S_2 \\ S_1 &\to XC \\ C &\to cC \mid \epsilon \\ X &\to aXb \mid \epsilon \\ S_2 &\to AY \\ A &\to aA \mid \epsilon \\ Y &\to bYc \mid \epsilon \end{aligned}$$

Now consider the empty string (ϵ). Since in this string $i = j$ and $j = k$ both hold, it can be derived with both rules $S \to S_1$ and $S \to S_2$. So this grammar is ambiguous.

1.3

1.3.1

Rightmost derivation of *bbab*:

$$S \Rightarrow ST \Rightarrow Sb \Rightarrow TSb \Rightarrow TTSb \Rightarrow TTab \Rightarrow Tbab \Rightarrow bbab \tag{1.3.1}$$

So the derivation tree is:

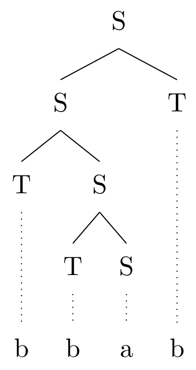

Figure 1: Rightmost derivation tree for *bbab*

1.3.2

$$S \Rightarrow TS \Rightarrow bS \Rightarrow bTS \Rightarrow bbS \Rightarrow bbST \Rightarrow bbaT \Rightarrow bbab \tag{1.3.2}$$

And the derivation tree is:

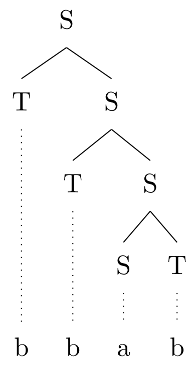

Figure 2: Leftmost derivation tree for *bbab*

1.3.3

As we saw, the string *bbab* had two different derivation trees (we can reach the tree in Fig. 1 by using leftmost derivations too. So this string has two leftmost derivations). Hence, this grammar is ambiguous. The reason behind this is that it has two choices in some steps to generate bs around the a.

1.3.4

We can convert it to a unambiguous grammar as follows:

$$\begin{aligned} S &\to TaT \\ T &\to bT \mid \epsilon \end{aligned}$$

1.4

Add new start variable:

$$\begin{aligned} S_0 &\to S \\ S &\to ASB \\ A &\to aASA \mid a \mid \epsilon \\ B &\to SbS \mid A \mid bb \end{aligned} \tag{1.4.1}$$

Remove $A \to \epsilon$:

$$\begin{aligned} S_0 &\to S \\ S &\to ASB \mid SB \\ A &\to aASA \mid a \mid aSA \mid aAS \mid aS \\ B &\to SbS \mid A \mid bb \mid \epsilon \end{aligned} \tag{1.4.2}$$

Remove $B \to \epsilon$:

$$\begin{aligned} S_0 &\to S \\ S &\to ASB \mid SB \mid S \mid AS \\ A &\to aASA \mid a \mid aSA \mid aAS \mid aS \\ B &\to SbS \mid A \mid bb \end{aligned} \tag{1.4.3}$$

Remove $S \to S$, $S_0 \to S$, and $B \to A$:

$$\begin{aligned} S_0 &\to ASB \mid SB \mid AS \\ S &\to ASB \mid SB \mid AS \\ A &\to aASA \mid a \mid aSA \mid aAS \mid aS \\ B &\to SbS \mid A \mid bb \mid aASA \mid a \mid aSA \mid aAS \mid aS \end{aligned} \tag{1.4.4}$$

And finally, we convert all the remaining rules into the proper form.

$$\begin{aligned} S_0 &\to AX \mid SB \mid AS \\ X &\to SB \\ S &\to AX \mid SB \mid AS \\ A &\to UY \mid a \mid UZ \mid UW \mid US \\ U &\to a \\ Y &\to AZ \\ Z &\to SA \\ W &\to AS \\ B &\to SK \mid VV \mid UY \mid a \mid UZ \mid UW \mid US \\ K &\to VS \\ V &\to b \end{aligned} \tag{1.4.5}$$

1.5 To Do

2 Closure Properties of Context-Free Languages

2.1

Suppose that L_1 and L_2 are two context-free languages. So there are CFGs that describe these languages, e.g. G_1 and G_2. Then we can create a CFG that describes the concatenation of these languages as follows:

$$\begin{aligned} &G_1 = (V_1, \Sigma, R_1, S_1)\\ &G_2 = (V_2, \Sigma, R_2, S_2)\\ \implies &G_3 = (V_1 \cup V_2 \cup \{S\}, \Sigma, R_1 \cup R_2 \cup \{S \to S_1S_2\}, S) \end{aligned}$$

And G_3 describes the concatenation of L_1 and L_2. So the concatenation of two context-free languages is also context-free.

2.2

First we convert the grammar of L to the Chomsky normal form. Then we convert this grammar (G) to a new grammar that recognizes the language $PREFIX(L)$, proving that $PREFIX(L)$ is context-free.

$$\begin{aligned} &G = (V, \Sigma, R, S)\\ \implies &G_2 = (V_2, \Sigma, R_2, S_\epsilon)\\ \text{Where } &V_2 = V \cup \left[\bigcup_{A\in V} \{A_\epsilon\}\right]\\ &R_2 = R \cup \left[\bigcup_{A\to BC\in R} \{A_\epsilon \to BC_\epsilon, A_\epsilon \to B_\epsilon\}\right] \cup \left[\bigcup_{A\to a\in R} \{A_\epsilon \to a\}\right] \cup \left[\bigcup_{A\in V} \{A_\epsilon \to \epsilon\}\right] \end{aligned}$$

For each A in the variables, we add a new variable A_ϵ, where A_ϵ is nullable. Then for each rule $A \to BC$, we add two new rules: $A_\epsilon \to BC_\epsilon$ and $A_\epsilon \to B_\epsilon$. And also for we copy all rules generating terminals (like $A \to a$) for the new variables too ($A_\epsilon \to a$). And we set S_ϵ to be the start variable. This way, if you consider the leftmost derivation of any string in L, if we start from S_ϵ instead of S, then at each step of the derivation there is a ϵ variable at the end of the string, where we can end the derivation right there by using the epsilon rule of that variable to get one of the strings in $PREFIX(L)$, or act as before and proceed further.

3 Pushdown Automata

3.1

3.1.1

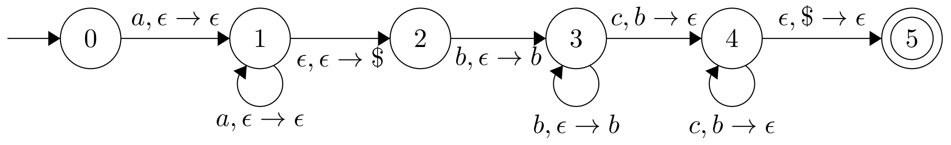
0
1
2
3
4
5
$a, \epsilon \to \epsilon$
$\epsilon, \epsilon \to \$$
$b, \epsilon \to b$
$c, b \to \epsilon$
$\epsilon, \$ \to \epsilon$
$a, \epsilon \to \epsilon$
$b, \epsilon \to b$
$c, b \to \epsilon$

Figure 3: Pushdown automata for the language $L = \{a^n b^m c^m \mid n, m \in \mathbb{N}\}$.

3.1.2

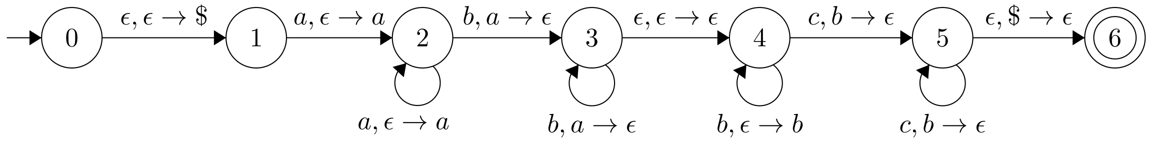

Figure 4: Pushdown automata for the language $\left\{a^i b^j c^k \mid i, j, k \in \mathbb{N}, i + k = j\right\}$

3.1.3

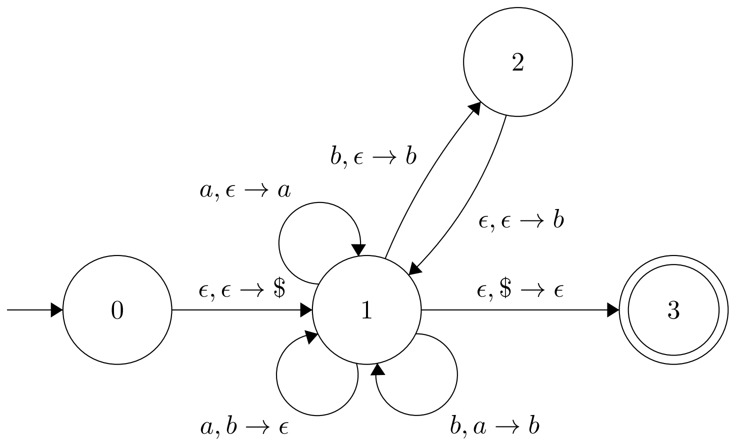
0
1
2
3
$b, \epsilon \to b$
$a, \epsilon \to a$
$\epsilon, \epsilon \to b$
$\epsilon, \epsilon \to \$$
$\epsilon, \$ \to \epsilon$
$a, b \to \epsilon$
$b, a \to b$

Figure 5: Pushdown automata for the language $\{w \in \{a, b\}^* \mid n_a(w) = 2n_b(w)\}$

Reading each b either pushes two bs to the stack or pops an a and adds a b to the stack. Reading each a either pushes and a to or pops a b from the stack.

3.2

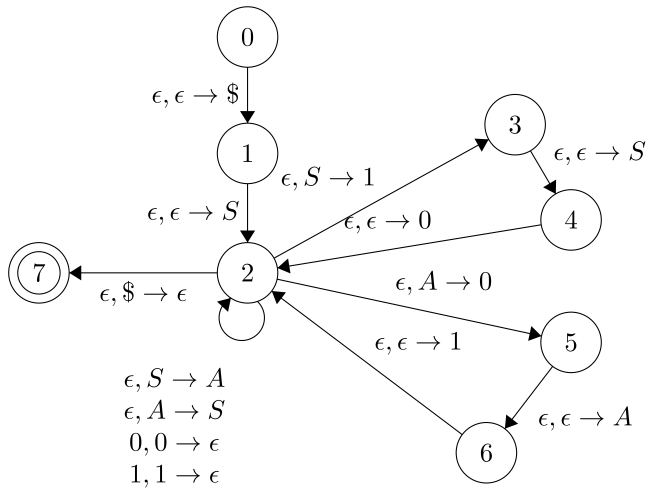
0
1
2
3
4
5
6
7
$\epsilon, \epsilon \to \$$
$\epsilon, \epsilon \to S$
$\epsilon, S \to 1$
$\epsilon, \epsilon \to S$
$\epsilon, \epsilon \to 0$
$\epsilon, \$ \to \epsilon$
$\epsilon, A \to 0$
$\epsilon, \epsilon \to 1$
$\epsilon, \epsilon \to A$
$\epsilon, S \to A$
$\epsilon, A \to S$
$0, 0 \to \epsilon$
$1, 1 \to \epsilon$

Figure 6: Pushdown automata for the grammar.

3.3 To Do

4 Pumping Lemma for Context-Free Languages

4.1

Suppose it holds. Consider the language $L = \left\{a^n b^m c^m d^{3n} \mid m, n \geq 1\right\}$. L is context-free, since there is a CFG describing it:

$$\begin{aligned} S &\to aSddd \mid aTddd \\ T &\to bTc \mid bc \end{aligned}$$

So $L_{1/2}$ will be as follows:

$$L_{1/2} = \begin{cases} a^n b^m c^n & m \geq n \\ a^n b^m c^m d^{n-m} & m < n \end{cases} \qquad m, n \geq 1$$

Suppose its pumping length is p. Let $s = a^p b^p c^p$ and $s \in L_{1/2}$. As its length is more than p, $a^p b^p c^p$ must be pumped. But we can show it cannot be pumped.

$$\exists u, v, x, y, z : \begin{cases} a^p b^p c^p = uvxyz \\ uv^i xy^i z \in L_{1/2} \quad i \geq 0 \\ |vxy| \leq p \\ |vy| > 0 \end{cases}$$

Consider two cases below:

- Either v or y contains more than one type of symbol, the order of symbols in uv^2xy^2z will not be in the correct order.
- Both v and y have only one type of alphabet. There can be 3 cases:
  - v and y both contain only a or both contain only c. In this case, number of as and cs in uv^2xy^2z will not be equal and it will not be in the language.
  - v and y both contain only b. In this case, uxz will have less bs than as and cs. Therefore it isn't a member of $L_{1/2}$.
  - v contains a and y contains b, or v contains b and y contains c. In this both of these cases, uv^2xy^2z will have different number of a and c.

So we've shown that $a^pb^pc^p$ cannot be pumped and reached a contradiction. Thus the mentioned rule does not hold.

4.2

We first assume these languages are context free, then we prove that they can't be pumped.

4.2.1

$$L = \left\{w \in \{a, b, c\}^* \mid n_a(w) < n_b(w) < n_c(w)\right\}$$

Consider the string $a^pb^{p+1}c^{p+2}$.

$$\exists u, v, x, y, z : \begin{cases} a^pb^{p+1}c^{p+2} = uvxyz \\ uv^ixy^iz \in L & i \geq 0 \\ |vxy| \leq p \\ |vy| > 0 \end{cases}$$

There are three cases for v and y:

- Either v or y contains more than one type of symbol, the order of symbols in uv^2xy^2z will not be in the correct order.
- Both v and y have only one type of alphabet and both contain the same symbol.
  - If both contain only a, then the number of a symbols in uv^2xy^2z, will be greater than or equal to number of b symbols.
  - If both contain only b, then the number of b symbols in uv^2xy^2z, will be greater than or equal to number of c symbols.
  - If both contain only c, then the number of c symbols in uxz, will be less than or equal to number of b symbols.
- Both v and y have only one type of alphabet and contain different symbols. There can be two cases:
  - v contains a and y contains b.

    - If y is not empty, then the number of b symbols in uv^2xy^2z, will be greater than or equal to number of c symbols.
    - If y is empty, then the number of a symbols in uv^2xy^2z, will be greater than or equal to number of b symbols.
  - v contains b and y contains c.
    - If v is not empty, then the number of b symbols in uxz will be less than or equal to number of a symbols.
    - If v is empty, then the number of c symbols in uxz will be less than or equal to number of b symbols.

So we've shown that $a^pb^{p+1}c^{p+2}$ cannot be pumped and reached a contradiction.

4.2.2

$$L = \left\{w \in \{a,b\}^* \mid w = w^R, n_a(w) = n_b(w)\right\}$$

Consider the string $s = a^pb^{2p}a^p$.

$$\exists u,v,x,y,z : \begin{cases} a^pb^{2p}a^p = uvxyz \\ uv^ixy^iz \in L & i \geq 0 \\ |vxy| \leq p \\ |vy| > 0 \end{cases}$$

- If either v or y contains more than one type of symbol, the order of symbols in uv^2xy^2z will not be in the correct order.
- Both v and y have only one type of alphabet and both contain the same symbol. In this case uv^2xy^2z, the number of a and bs will be different.
- v contains a and y contains b or vice versa. In this case, uv^2xy^2z will not be symmetric.

So we've shown that $a^pb^{2p}a^p$ cannot be pumped and reached a contradiction.

4.2.3

$$L = \left\{a^ib^j \mid i = kj \text{ for some positive integer } k\right\}$$

Consider the string $s = a^{2p^2}b^{2p}$. As $|s|$ is greater than p, it can be pumped:

$$\exists u,v,x,y,z : \begin{cases} a^{2p^2}b^{2p} = uvxyz \\ uv^ixy^iz \in L & i \geq 0 \\ |vxy| \leq p \\ |vy| > 0 \end{cases}$$

There are two cases for v and y:

- Either v or y contains more than one type of symbol. In this case, the order of symbols in uv^2xy^2z will not be in the correct order.
- Both v and y have only one type of alphabet and both contain the same symbol.
  - Both contain a. Suppose $|vy| = n$. We have:

$$\begin{aligned} & \forall i : 2p \mid 2p^2 + (i-1)n \\ \Longrightarrow\ & \forall i : 2p \mid (i-1)n \\ & \Longrightarrow 2p \mid n \\ & \Longrightarrow n \geq 2p \end{aligned}$$

    Which is a contradiction. Because we know $n = |vy| \leq |vxy| \leq p$.
  - Both contain b. In this case we can find a i large enough such that uv^ixy^iz contains more bs than as.
  - v contains a and y contains b. Assume $|v| = n$ and $|y| = m$. We have:

$$\begin{aligned} & \forall i : 2p + (i-1)m \mid 2p^2 + (i-1)n \\ \Longrightarrow\ & \forall i : 2p + (i-1)m \mid (i-1)(n - pm) \\ & \Longrightarrow n - pm = 0 \\ & \Longrightarrow n = pm \\ & \Longrightarrow |vy| = (p+1)m > p \end{aligned}$$

    Which is again, a contradiction because we know $|vy| \leq p$.

So we've shown that $a^{2p^2}b^{2p}$ cannot be pumped and reached a contradiction.