

CSE2315 — Assignment 2

Vlad Paun
6152937

February 25, 2026

Exercise 1

Suppose we have the following language L over the alphabet $\Sigma = \{a, 0, 1\}$:

$$L = \left\{ ua^k v \mid u, v \in \{0, 1\}^* \text{ and } c_0(u) + c_1(v) > k \right\}$$

Where $c_x(w)$ is defined as the number of occurrences of x in w . Use the pumping lemma to show that L is not regular.

Proof. By contradiction.

Assume to the contrary that L is regular. Let p be the pumping length given by the pumping lemma. Choose s to be the string $a^p 1^p 1$. Because s is a member of L and s has length more than p , the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string $xy^i z$ is in L . We show this outcome is impossible.

Condition 3 of the pumping lemma says that $|xy| \leq p$. Thus, in the case of our word, y must consist of only a s. Let k_{xyz} and k_{xyyz} represent the number of a s in xyz and $xyyz$ respectively. $k_{xyz} = p$ and $k_{xyyz} > k_{xyz} = p$, but the number of 1s in $xyyz$ is still $p + 1$. Thus in $xyyz$ we have $c_0(u) + c_1(v) \leq k_{xyyz}$, so $xyyz$ cannot be in L . Therefore, s cannot be pumped and we have reached our desired contradiction. \square

Exercise 2

Suppose we have the context free grammar $G = (\{S, T, V\}, \{a, b, c\}, R, S)$, with R containing the following rules:

$$\begin{aligned} S &\rightarrow Tc \mid cV \\ T &\rightarrow aTb \mid bTa \mid \varepsilon \\ V &\rightarrow a \mid b \end{aligned}$$

- (a) Describe the set $L(G)$ in your own words.
- (b) Construct a PDA M such that $L(M) = L(G)$. Use no more than 8 states.
- (c) Explain how your PDA works.

Exercise 3

Consider the following rules R of a CFG $G = (\{S, A, B\}, \{a, b\}, R, S)$:

$$\begin{aligned} S &\rightarrow bB \mid abB \\ A &\rightarrow bA \mid AA \\ B &\rightarrow AS \mid SA \mid \epsilon \end{aligned}$$

Convert G into Chomsky normal form.

First, we add a new start variable.

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow bB \mid abB \\ A &\rightarrow bA \mid AA \\ B &\rightarrow AS \mid SA \mid \epsilon \end{aligned}$$

Then, we remove the ϵ -rules.

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow bB \mid abB \mid \mathbf{b} \mid \mathbf{ab} \\ A &\rightarrow bA \mid AA \\ B &\rightarrow AS \mid SA \mid \epsilon \end{aligned}$$

Third, we remove all unit rules.

$$\begin{aligned} S_0 &\rightarrow S \mid \mathbf{bB} \mid \mathbf{abB} \mid \mathbf{b} \mid \mathbf{ab} \\ S &\rightarrow bB \mid abB \mid b \mid ab \\ A &\rightarrow bA \mid AA \\ B &\rightarrow AS \mid SA \end{aligned}$$

Convert the remaining rules into proper form.

$$\begin{aligned} S_0 &\rightarrow VB \mid TB \mid b \mid UV \\ S &\rightarrow VB \mid TB \mid b \mid UV \\ A &\rightarrow VA \mid AA \\ B &\rightarrow AS \mid SA \\ T &\rightarrow UV \\ U &\rightarrow a \\ V &\rightarrow b \end{aligned}$$

Exercise 4

Let L be the set of all TUDON words.

- (a) Give a CFG G such that $L(G) = L$, using no more than 6 variables and 9 + 26 rules.

Let $G = (\{S, T, U, V, X\}, \{<, >, :, ;, a, b, \dots, z\}, R, S)$, with R containing the following rules:

$$\begin{aligned} S &\rightarrow < T > \\ T &\rightarrow U : V X \\ X &\rightarrow ; T \mid \varepsilon \\ U &\rightarrow \Sigma U \mid \varepsilon \\ V &\rightarrow S \mid U \end{aligned}$$

Where Σ refers to any character a, b, \dots, z

- (b) Explain why your grammar generates L

So basically the start variable makes one of those dictionaries, and then defers to a new variable for the contents of the dict. Then that variable T makes a key value pair, appended with an X. Then the X can be nothing, or a semicolon and then another T. And then the actual key is any number of alphabet characters, and the value is also any number of alphabet characters or another dictionary.

- (c) Is your grammar ambiguous? Give an example or an argument why not.

The grammar is not ambiguous, the only part where it could be ambiguous, is when creating the key-value pairs in the dictionary, you could substitute them and creating them in different orders, but I take care of that by appending them one by one do what is already there. This way, there is no ambiguous word.

Exercise 5

Let the operator @ be defined as $A@ = \{a^{|w|} \mid w \in A\}$. Prove that regularity is closed under @.

Proof. Let A be a regular language. We will prove that $A@$ is also regular.

Since A is regular, there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = A$. Now we will build the NFA $M' = (Q, \{a\}, \Delta, q_0, F)$, where $\Delta(q, a) = \{\delta(q, x) \mid x \in \Sigma\}$. In other words, take M and replace every symbol on transition arrows with a .

Next, we must show that the set of words reachable in M' by reading n as from the input is equivalent (meaning the same lengths of words) to the set of words reachable in M by reading any n characters from an input word. This can be shown using induction. It is left as an exercise to the reader. \square

Exercise 6

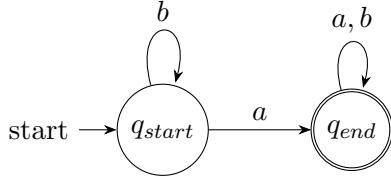
- (a) Let N be an arbitrary NFA with k states. Explain why if $L(N) \neq \emptyset$, then there is a word of length $\leq k$ in $L(N)$.

If $L(N) \neq \emptyset$, then N must recognize some word. This means that there must be a path from the starting state to some accepting state given by a set of transitions. More formally, take any

$w \in L(N)$, where $w = w_1 w_2 \dots w_m$. There must be a sequence of states r_0, r_1, \dots, r_m , such that r_0 is the starting state, r_m is an accepting state, and for all $0 \leq i < m$, $r_{i+1} \in \delta(r_i, w_{i+1})$. Now, if $m \leq k$, we are done. However, if $m > k$, then by the pigeonhole principle some state must be repeated. So, in the sequence $r_0, \dots, r_x, \dots, r_x, \dots, r_m$, we can remove everything between the r_x s, including one of the r_x s themselves, and we can keep doing this procedure until the length of the sequence becomes $\leq k$. Now, this sequence exactly describes an input word of length $\leq k$ that is in $L(N)$.

- (b) Let D be an arbitrary DFA with $k > 1$ states. Give a counterexample to the claim that there is a DFA D' with $k - 1$ states, such that $L(D') = L(D)$.

Let D be the DFA described by the following state diagram (assume the alphabet is $\{a, b\}$):



This DFA only recognizes words that contain at least one a inside. If we were to construct a DFA with $k - 1$ states, it would have to have 1 state. This state could either be accepting, or not. Which means D' would either accept every word, or no words. Neither of which are equivalent to D .

Bonus Exercise