Homework 4 Solution

Formal Languages and Automata (CS322)

Last update: December 11, 2025

1. Consider the context-free grammar $G = (\{S, T, W, U, X, A, B\}, \Sigma, R, S)$ over $\Sigma = \{a, b\}$, where the rules R are as follows.

$$\begin{array}{rcl} S & \to & TW \mid AT \mid TA \mid a \\ T & \to & AU \mid BX \mid TT \mid AB \mid BA \\ W & \to & AT \\ U & \to & TB \\ X & \to & TA \\ A & \to & a \\ B & \to & b \end{array}$$

Determine whether the following strings are in $L(G)$ by using the CYK algorithm. Give a table constructed by the algorithm for each string.

(a) $ababa$

(b) $baabaa$

Solution.

(a) The string is in $L(G)$.

$\{S, W, X\}$	$\{T\}$	$\{S, W, X\}$	$\{T\}$	$\{S, A\}$
$\{T\}$	$\{U\}$	$\{T\}$	$\{B\}$	
$\{S, W, X\}$	$\{T\}$	$\{S, A\}$		
$\{T\}$	$\{B\}$			
$\{S, A\}$				
a	b	a	b	a

(b) The string is not in $L(G)$.

-	-	-	$\{S, X\}$	-	$\{S, A\}$
$\{S, X\}$	-	$\{S, W, X\}$	$\{T\}$	$\{S, A\}$	
$\{T\}$	$\{S, W\}$	$\{T\}$	$\{B\}$		
$\{S, X\}$	-	$\{S, A\}$			
$\{T\}$	$\{S, A\}$				
$\{B\}$					
b	a	a	b	a	a

$\square$

2. Use the pumping lemma for CFLs to show that the following languages are not context-free.

(a) $L_3 = \{w \in \Sigma^* : |w| \text{ is a prime number.}\}$ for any nonempty finite Σ

(b) $L_4 = \{w \in \{a,b\}^* : w = a^n b^n a^n b^n \text{ for some } n \in \mathbb{N}\}$

Solution.

(a) Let a be an element of Σ. For every positive number $\forall p$, consider $\exists w = a^k \in L_3$ where $k > p$ is prime number. For every proper split $uvxyz = w$ with $|vy| \geq 1$ and $|vxy| \leq p$, let $|vy| = t$. Then, $uv^ixy^iz = a^{k+(i-1)t}$ for every $i \in \mathbb{N}$. Then, consider uv^jxy^jz where $j = k+1$. Then, $uv^ixy^iz = a^{k+kj} = a^{k(j+1)}$. $k(j+1)$ is not a prime number since $k > p \geq 1$ divides it. So, $uv^jxy^jz \notin L_3$. It means that there exists $j \geq 0$ with $uv^jxy^jz \notin L_3$. Therefore, L_3 is not context-free by contrapositive of the pumping lemma for CFLs.

(b) For every positive number $\forall p$, consider $\exists w = a^pb^pa^pb^p \in L_4$. For every proper split $uvxyz = w$ with $|vy| \geq 1$ and $|vxy| \leq p$, consider $uv^0xy^0z = uxz$.

Assume that $uxz \in L_4$. Since $|uxz| = |w| - |vy| < |w|$, $|uxz| = a^mb^ma^mb^m$ for some $m < p$. The first p symbols of uxz are a^mb^{p-m}, and they are different from first p symbols of $w = uvxyz$. It means that vy overlaps with the first p symbols of w. By same logic, vy overlaps with the last p symbols of w. Therefore, vxy fully include b^pa^p, which is substring of w. ($w = a^p(b^pa^p)b^p$). It contradicts with the rule of proper split, $|vxy| \leq p$. $\Rightarrow\Leftarrow$

So, $uv^0xy^0z \notin L_4$. It means that there exists $i = 0 \geq 0$ with $uv^ixy^iz \notin L_4$. Therefore, L_4 is not context-free by contrapositive of the pumping lemma for CFLs.

□

3. Answer the following. If the problem asks you to use properties of CFLs, you need to prove it without using the pumping lemma for CFLs.

(a) Show, by using the pumping lemma, that the language $L_6 = \{\mathtt{0}^n\mathtt{1}^{2n}\mathtt{0}^n : n \geq 0\}$ is not context-free.

(b) Show that the language $L_7 = \{w \in \{\mathtt{0},\mathtt{1}\}^* : w = w^{\mathcal{R}} \text{ and } |w|_\mathtt{0} = |w|_\mathtt{1}\}$ is not context-free by using the part (a) and properties of CFLs introduced in the lecture.

(c) Show that $L_2 = \{w \in \{a,b,c\}^* : |w|_a = |w|_b = |w|_c\}$ is not context-free by using the properties of CFLs. You may use any languages shown (or not shown) to be context-free in the lecture without providing a proof, but you must indicate that they were discussed in the lecture.

Solution.

(a) Assume L_6 is context-free. Let p be the pumping length from the lemma. Take the string $w = 0^p1^{2p}0^p \in L_6$. Clearly $|w| \geq p$, so the lemma should apply. By the lemma, we should be able to write $w = uvxyz$ with $|vy| \geq 1$, $|vxy| \leq p$, and $uv^ixy^iz \in L_6$ for all $i \geq 0$. Now, split w into three blocks: 0^p (first), 1^{2p} (second), and 0^p (third). Note that vxy can intersect at most one boundary between blocks. So, placement of vxy can be one of the following:

(1) entirely inside the first block,
(2) entirely inside the middle block,
(3) entirely inside the last block,
(4) crossing between the first and the second block, and
(5) crossing between the second and the third block.

Observe that in the first three cases, pumping down (setting $i = 0$) takes us outside of L_6, a contradiction to the pumping lemma. Case 4 and 5 are symmetric, so we will only talk about Case 4. In that case, again pumping down decreases the number of 0s in the first block or decreases the number of 1s in the second block (note: both may happen). Consequently, the number of 0s and 1s in the first and the second block, respectively, do not match what they should be had the new word been in L_6, given that the number of 0s in the third block stays p. We again arrive at a contradiction to the pumping lemma. Thus, our initial assumption that L_6 is a CFL was wrong.

(b) Pick the regular language $R := 0^*1^*0^*$. Note that $L_7 \cap R = L_6$. In the lectures, we discussed that CFLs are closed under taking intersection with a regular language. So, if L_7 were CFL, $L_7 \cap R = L_6$ would have been a CFL as well. But this is false by (a). Hence L_7 cannot be a CFL.

(c) Recall: we proved in the lecture that the language $A := \{a^nb^nc^n : n \geq 0\}$ is not a CFL. Now, consider the regular language $R := a^*b^*c^*$. Observe that $L_2 \cap R = A$. If L_2 where a CFL, $L_2 \cap R = A$ would have been a CFL as well, which is not true. So, L_2 cannot be a CFL.

□

4. For a language L, define $\mathsf{SUFFIX}(L) := \{v : uv \in L \text{ for some string } u\}$ and recall $\mathsf{HALVES}(L) := \{w \in \Sigma^* : \text{ there is } u \in \Sigma^* \text{ such that } |u| = |w| \text{ and } wu \in L\}$.

(a) Show that the set of context-free languages is closed under the SUFFIX operation. If you propose a PDA or a CFG for $\mathsf{SUFFIX}(L)$ based on a CFG or a PDA for L, provide a complete description of how the PDA/CFG is constructed; however, you are not required to provide a proof of the proposed CFG or PDA recognizing $\mathsf{SUFFIX}(L)$.

(b) Show that the set of context-free languages is **not** closed under the HALVES operation.

Solution.

(a) Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA with $L(M) = L$. We build a PDA M' such that $L(M') = \mathsf{SUFFIX}(L)$.

Intuitively, M' has two modes, encoded by a bit in the state:

- mode 0: "skipping" an arbitrary prefix u of the input without consuming the input word we want to test, and
- mode 1: simulating M on the actual suffix w.

Formally, define $M' = (Q', \Sigma, \Gamma, \delta', q_0', F')$, where $Q' = Q \times \{0, 1\}$, $q_0' = (q_0, 0)$, $F' = \{(q, 1) \mid q \in F\}$, and the stack alphabet Γ is unchanged.

We describe δ' by cases. Write states of M' as (q, b) with $q \in Q$ and $b \in \{0, 1\}$.

- In mode 0, we never consume real input symbols; we only use ε-moves to mimic any sequence of moves of M on some hypothetical prefix u. For all $q \in Q$, $r \in \Gamma_\varepsilon$, and for every transition $(p, s) \in \delta(q, a, r)$ of M with $a \in \Sigma_\varepsilon$, we put $\delta'((q, 0), \varepsilon, r) \ni ((p, 0), s)$. This part says: $(q, 0)$ can follow any move that M could make at q, without consuming the actual input of M'.
- We also let the PDA M' non-deterministically guess that it finished mode 0 and will move on to mode 1. At any time we may guess that we have reached the boundary between u and w, and start simulating M on the real input. For all $q \in Q$ we add $\delta'((q, 0), \varepsilon, \varepsilon) \ni ((q, 1), \varepsilon)$. This does not touch the stack and simply flips the mode bit from 0 to 1.
- Now we describe mode 1 part, where we actually simulate M. Once in mode 1, M' behaves exactly like M on the remaining input. For all $q \in Q$, $a \in \Sigma_\varepsilon$, $r \in \Gamma_\varepsilon$ and every $(p, s) \in \delta(q, a, r)$, set $\delta'((q, 1), a, r) \ni ((p, 1), s)$. Note that the mode bit remains 1, so we never return to the "skipping" phase.

We accept exactly when the mode-1 segment ends in a state $(q, 1)$ with $q \in F$. This M' recognizes $\mathsf{SUFFIX}(L)$.

(b) Let $\Sigma = \{a, b, \texttt{\#}\}$ and define $L = \{\, a^n b^m a^m \texttt{\#\#} b^{3n} : n, m \geq 0 \,\}$. The grammar

$$S \to aSb^3 \mid X, \qquad X \to Y\texttt{\#\#}, \qquad Y \to bYa \mid \varepsilon$$

generates exactly L, so L is context-free.

Assume, for contradiction, that context-free languages are closed under HALVES. Then $L' := \mathsf{HALVES}(L)$ should also be context-free.

Let $R = a^*b^*a^*\texttt{\#}$, which is regular. Since CFLs are closed under intersection with regular languages, $L' \cap R$ should be context-free.

Take any $x \in L' \cap R$. Because $x \in R$, it has the form $x = a^i b^j a^k \texttt{\#}$. Because $x \in L'$, there exist n, m and a string u such that $xu = a^n b^m a^m \texttt{\#\#} b^{3n}$ and $|x| = |u|$. So, $x = a^n b^m a^m \texttt{\#}$ and $u = \texttt{\#} b^{3n}$ for some $n, m \geq 0$. From the equality of their lengths, we have that $n + 2m + 1 = 1 + 3n$, which implies $n = m$. Hence $L' \cap R = \{a^n b^n a^n \texttt{\#} : n \geq 0\} =: L''$.

We further intersect L'' with $a^*b^*a^*$ to obtain $L''' := \{a^n b^n a^n : n \geq 0\}$. Should L'' be context-free, L''' must be too, but one can show, for example using the pumping lemma, that L''' is not context-free.

Therefore $L' \cap R$ is not context-free, so our assumption that $L' = \mathsf{HALVES}(L)$ is context-free is false. Consequently, context-free languages are not closed under the HALVES operation.

$\square$

5. In this problem, we introduce *Ogden's lemma*, a generalization of the pumping lemma for CFLs.

Lemma 1 (Ogden's lemma)**.** *If a language L is generated by a context-free grammar $G = (V, \Sigma, R, S)$, then there is an integer $p \geq 1$ for which the following holds. For every string $s \in L$ of length at least p and every way of marking at least p positions of s as distinguished, there is a nonterminal $A \in V$ and a rewriting of s as $s = uvwxy$ such that*

(O1) $S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy$;

(O2) the string w contains at least one marked position;

(O3) the string vwx contains at most p marked positions; and

(O4) u, v both contain marked positions, or x, y both contain marked positions.

Answer the following questions.

(a) Justify that Ogden's lemma implies the following lemma:

Lemma 2. *For every context-free language L over an alphabet Σ, there is an integer $p \geq 1$ such that the following holds. For every string $s \in L$ of length at least p and every way of marking at least p positions of s as distinguished, there is a rewriting of s as $s = uvwxy$ such that*

(L1) the string vx contains at least one marked position;

(L2) the string vwx contains at most p marked positions; and

(L3) $uv^nwx^ny \in L$ for every $n \in \mathbb{N}$.

Based on this, explain why Ogden's lemma is a generalization of the pumping lemma for CFLs.

(b) By using Ogden's lemma, prove that the language

$$L_1 := \{\mathsf{a}^i\mathsf{b}^j\mathsf{c}^j\mathsf{d}^j \mid i, j \in \mathbb{N}\} \cup \{\mathsf{b}^j\mathsf{c}^k\mathsf{d}^\ell \mid j, k, \ell \in \mathbb{N}\}$$

over the alphabet $\{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ is not context-free.

Solution.

(a) Let $p \geq 1$ be an integer satisfying Ogden's lemma, and let $s \in L$ be a string of length at least p. We mark every position of s as distinguished. Then:

- (L1) is deduced by (O4);
- (L2) is deduced by (O3); and
- (L3) is deduced by (O1) since it implies that $A \to vAx \mid w$ is in R.

(b) We use Lemma 2 to show that L_1 is not context-free. Suppose to the contrary that there is an integer $p \geq 1$ satisfying Lemma 2. Take $s = \mathsf{a}\mathsf{b}^p\mathsf{c}^p\mathsf{d}^p \in L_1$ and let all b's are marked as distinguished. Then there is a rewriting $s = uvwxy$ that satisfies (L1)-(L3). Observe that if v or x contains more than one symbol, then it does not satisfy (L3). Thus, v and x consist of exactly one symbol. Moreover, by (L1), at least one of v, x consists of only b. This shows that we have the following possible cases:

- $v \in \{\epsilon, \mathsf{a}\}$ and $x = \mathsf{b}^i$ for some $i \geq 1$;
- $v = \mathsf{b}^i$ and $x = \mathsf{b}^j$ for some $i, j \geq 0$ satisfying $1 \leq i + j \leq p$;
- $v = \mathsf{b}^i$ and $x = \mathsf{c}^j$ for some $i \geq 1$ and $j \geq 0$; and
- $v = \mathsf{b}^i$ and $x = \mathsf{d}^j$ for some $i \geq 1$ and $j \geq 0$.

However, in any cases, we have $uv^2wx^2y \notin L_1$ and we reach a contradiction. Therefore, the language L_1 is not context-free.

□

6. A *queue* is a tape that allows symbols to be written only at the left-hand end and read only at the right-hand end. A *pushdown automaton* $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ *with a queue* is similar to a PDA, but it uses a queue instead of a stack. The automaton M has a separate, read-only input tape in addition to the queue. The input tape head can only move from left to right, one cell at a time, just as a PDA. There is a blank symbol in the cell following the input, so the end of the input can be detected.

The transition $(q, y) \in \delta(p, a, x)$ of M is interpreted as the following.

- Read the symbol a from the input tape.
- Update the current state p to q.
- Remove the symbol x at the right-hand end of the queue (called dequeue).
- Add the symbol y to the left-hand end of the queue (called enqueue).

Here, the symbols a, x, and y can be ϵ.

Prove that every **deterministic single-tape** Turing machine is equivalent to a pushdown automaton with a queue. A high-level description of the machines is sufficient for the proof, but you need to provide the description in enough detail so that a formal description can be easily deduced from it.

Solution. We first show that for every pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with a queue, there is a deterministic 2-tape TM $M' = (Q, \Sigma, \Gamma', \delta', q'_0, q'_{accept}, q'_{reject})$ that simulates M. As every multi-tape TM has an equivalent single-tape TM, this proves one direction of the statement.

We use the first tape as a read-only input tape and the other as a queue. The tape alphabet Γ' of M' contains every symbol in Γ and a new distinct symbol $\rhd$, which marks the left-hand end of the queue. The TM first writes $\rhd$ on the second tape. For a dequeue, the TM M' finds the first symbol $x \in \Gamma$ on the right-hand side of $\rhd$ and replaces it with $\rhd$. For an enqueue, starting from $\rhd$ going right, the TM M' finds the first blank symbol $\sqcup$ and replaces it with y.

Now, we show that for every deterministic single-tape TM $M' = (Q', \Sigma, \Gamma', \delta', q'_0, q'_{accept}, q'_{reject})$, there is a pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with a queue that simulates M'. First, M reads the input and enqueue every symbol one-by-one into the queue, and marks the end of the input with a new special symbol $\lhd$ in Γ', distinct from every symbol in Γ. We want to keep the left-hand end symbol in the queue to be the symbol under the the head of M'. Every transition of M' is simulated by M as follows.

- If the head moves to the right, i.e., we have $\delta'(q, a) \in (p, b, \mathsf{R})$, dequeue a from the queue and enqueue b to the queue. If M encounters $\rhd$ and has to move to the right, it does not dequeue $\rhd$.
- If the head moves to the left, i.e., we have $\delta'(q, a) \in (p, b, \mathsf{L})$, dequeue a from the queue and enqueue b to the queue as before. But now we need to move the symbol from the second position on the right end of the queue to the left end. In order to do so, the following shifting operation is conducted twice.

  For example, we have the following in the queue:

  $$x\, y \lhd z.$$

  One shifting operation is proceeded as follows.

  1. Enqueue a new symbol $\blacksquare$, which will mark the end of the queue. Then we have the following in the queue.
     $$x\, y \lhd z\, \blacksquare$$
  2. Replace every symbol s of the queue with a new symbol (s', s), where s' is the symbol on the (cyclically) left-hand side of s in the queue. This can be done by entering a new state, which is distinct from every state in Q and remembers the last symbol M read. Consider the example we have. First, M dequeues x, and enters the new state q_x. The following process is repeated until M dequeues $\blacksquare$. The automaton M dequeues y, enqueues (x, y) and enters the new state q_y. After finishing the process, we have the following in the queue.
     $$(x, y)\,(y, \lhd)\,(\lhd, z)\,(z, \blacksquare)$$
     Since there are finitely many input symbols and tape symbols, this process can be simulated by a (finite) pushdown automaton.

3. Repeatedly dequeue (s', s) and enqueue (s', s) until M encounters $\blacksquare$ in the second entry of the pair $(s', \blacksquare)$. Then M enqueues $\blacksquare$ and s'. After finishing this process, we have:

$$(x, y)\,(y, \lhd)\,(\lhd, z)\,\blacksquare\,z$$

4. Repeatedly dequeue (s', s) and enqueue s' until M dequeues $\blacksquare$, which concludes the whole process. Now, we have:

$$z\,x\,y\,\lhd$$

$\square$

7. Construct a single-tape deterministic Turing machine that decides the language

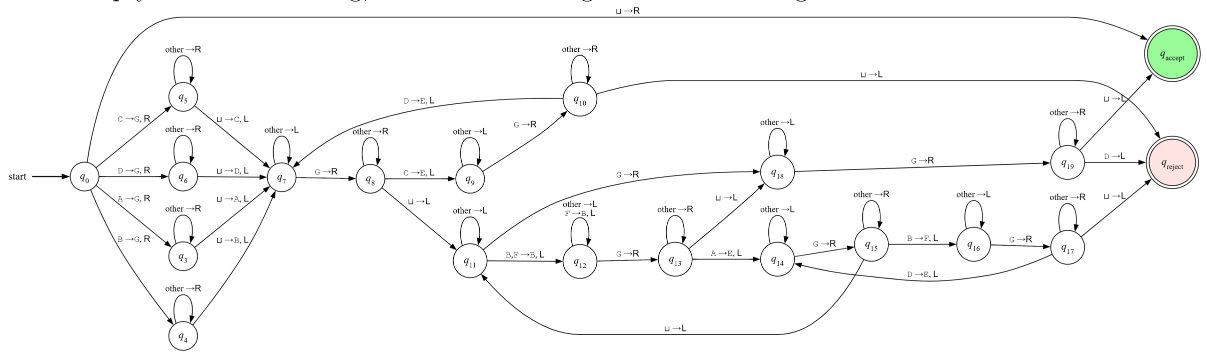
$$L = \{w \in \Sigma^* : |w|_{\texttt{A}} \cdot |w|_{\texttt{B}} + |w|_{\texttt{C}} = |w|_{\texttt{D}}\}$$

over the alphabet $\Sigma = \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$.

Solution. You can simply use (a variation of) the idea written in the last slide of the Lecture #14 to construct a Turing machine. The following is an example answer:

```
0 0 1 0 1
0 1 3 7 1
0 2 4 7 1
0 3 5 7 1
0 4 6 7 1
3 -1 3 -1 1
3 0 7 1 -1
4 -1 4 -1 1
4 0 7 2 -1
5 -1 5 -1 1
5 0 7 3 -1
6 -1 6 -1 1
6 0 7 4 -1
7 -1 7 -1 -1
7 7 8 -1 1
8 -1 8 -1 1
8 0 11 -1 -1
8 3 9 5 -1
9 -1 9 -1 -1
9 7 10 -1 1
10 -1 10 -1 1
10 0 2 -1 -1
10 4 7 5 -1
11 -1 11 -1 -1
11 2 12 2 -1
11 6 12 2 -1
11 7 18 -1 1
12 -1 12 -1 -1
12 6 12 2 -1
12 7 13 -1 1
13 -1 13 -1 1
13 0 18 -1 -1
13 1 14 5 -1
14 -1 14 -1 -1
14 7 15 -1 1
15 -1 15 -1 1
15 0 11 -1 -1
15 2 16 6 -1
16 -1 16 -1 -1
16 7 17 -1 1
17 -1 17 -1 1
17 0 2 -1 -1
17 4 14 5 -1
18 -1 18 -1 -1
18 7 19 -1 1
19 -1 19 -1 1
19 0 1 -1 -1
19 4 2 -1 -1
```

To help your understanding, the transition diagram of this Turing machine is as follows:



□

where $\sigma_5 = \texttt{E}$, $\sigma_6 = \texttt{F}$, $\sigma_7 = \texttt{G}$.