COMP 3803 — Fall 2024 — Solutions Assignment 3

Question 1: Write your name and student number.

Solution: Rodri, 16

Question 2: Consider the context-free grammar $G = (V, \Sigma, R, S)$, where the set of variables is $V = \{S, A, B\}$, the set of terminals is $\Sigma = \{a, b\}$, the start variable is S, and the rules are as follows:

$$\begin{array}{lcl} S & \to & aS \mid bA \\ A & \to & aA \mid bB \\ B & \to & \varepsilon \mid aB \end{array}$$

Determine the language $L(G)$ that is generated by G. Prove that your answer is correct. (Remember: To prove that two sets X and Y are equal, you have to prove that $X \subseteq Y$ and $Y \subseteq X$.)

Solution: After having stared at the rules for a very long time, it looks like the language $L(G)$ of this context-free grammar is

$$L = \{w \in \{a, b\}^* : \ w \text{ contains exactly two } b\text{'s}\}.$$

First we show that $L(G) \subseteq L$. Start with the start variable S. Every time we apply the rule $S \to aS$, we add one a to the string. At some point, we must apply the rule $S \to bA$. At this moment, we add the first b to the string. From then on, every time we apply the rule $A \to aA$, we add one a to the string. At some point, we must apply the rule $A \to bB$. At this moment, we add the second b to the string. From then on, every time we apply the rule $B \to aB$, we add one a to the string. At some point, we must apply the rule $B \to \varepsilon$ and terminate the derivation. This shows that every string in $L(G)$ has exactly two b's.

Next we show that $L \subseteq L(G)$. Any string w in L can be written as $w = a^k b a^\ell b a^m$, for some integers $k \geq 0$, $\ell \geq 0$, and $m \geq 0$. We can derive this string w in the following way:

- Start with S.
- k times, apply the rule $S \to aS$. This gives the string $a^k S$.
- Apply the rule $S \to bA$. This gives the string $a^k bA$.
- ℓ times, apply the rule $A \to aA$. This gives the string $a^k b a^\ell A$.
- Apply the rule $A \to bB$. This gives the string $a^k b a^\ell bB$.
- m times, apply the rule $B \to aB$. This gives the string $a^k b a^\ell b a^m B$.
- Apply the rule $B \to \varepsilon$. This gives the string $w = a^k b a^\ell b a^m$.

Question 3: Give context-free grammars that generate the following languages. For each case, justify your answer.
(3.1) $\{a^n b^m c^m d^{2n} : n \geq 0, m \geq 1\}$. The set of terminals is equal to $\{a, b, c, d\}$.
(3.2) $\{a^n b^m c^k : 0 \leq n + m \leq k\}$. The set of terminals is equal to $\{a, b, c\}$.
(3.3) The language of all strings over the alphabet $\{a, b\}$ that have the same number of a's as b's. The set of terminals is equal to $\{a, b\}$.

Solution: We start with

$$L_1 = \{a^n b^m c^m d^{2n} : n \geq 0, m \geq 1\}.$$

How can we "build" all strings in L_1:

- Start with the start variable S.
- Repeat the following zero or more times: Using the rule $S \to aSdd$, add one a and two d's. In this way, we can obtain all strings of the form $a^n S d^{2n}$, where $n \geq 0$.
- At some point, we will start adding b's and $c's$. Just before we do this, we apply the rule $S \to A$. Thus, we now have the strings $a^n A d^{2n}$ for any $n \geq 0$.
- Repeat the following zero or more times: Using the rule $A \to bAc$, add one b and one c. In this way, we can obtain all strings of the form $a^n b^{m-1} A c^{m-1} d^{2n}$, where $m \geq 1$.
- At the end, we use the rule $A \to bc$ to obtain the string $a^n b^m c^m d^{2n}$ for any $n \geq 0$ and $m \geq 1$.

This leads to the following context-free grammar:

$$\begin{array}{lcl} S & \to & aSdd \mid A \\ A & \to & bAc \mid bc \end{array}$$

It follows from the above explanation, that all strings in L_1 can be derived using these rules. It is clear that no other strings can be derived.

Next we do

$$L_2 = \{a^n b^m c^k : 0 \leq n + m \leq k\}.$$

How can we "build" all strings in L_2:

- Start with the start variable S.
- Repeat the following zero or more times: Using the rule $S \to aSc$, add one a and one c. In this way, we obtain all strings of the form $a^n S c^n$, where $n \geq 0$.
- At some point, we will start adding b's and c's. Just before we do this, we apply the rule $S \to B$. Thus, we now have the strings $a^n B c^n$ for any $n \geq 0$.

- Repeat the following zero or more times: Using the rule $B \to bBc$, add one b and one c. In this way, we obtain all strings of the form $a^n b^m B c^{n+m}$, where $n \geq 0$ and $m \geq 0$.
- At some point, we will start adding c's. Just before we do this, we apply the rule $B \to C$. Thus, we now have the strings $a^n b^m C c^{n+m}$, where $n \geq 0$ and $m \geq 0$.
- Repeat the following zero or more times: Using the rule $C \to Cc$, add one c. In this way, we obtain all strings of the form $a^n b^m C c^k$, where $k \geq n + m$.
- To remove the variable C, we apply the rule $C \to \varepsilon$.

This leads to the following context-free grammar:

$$\begin{array}{lcl} S & \to & aSc \mid B \\ B & \to & bBc \mid C \\ C & \to & \varepsilon \mid Cc \end{array}$$

It follows from the above explanation, that all strings in L_2 can be derived using these rules. It is clear that no other strings can be derived.

Finally, we do the language L_3 consisting of all strings that have the same number of a's as b's.

This one is a bit tricky[1]. Here is the context-free grammar:

$$S \to \varepsilon \mid aSbS \mid bSaS$$

It is clear that every string in the language of this context-fee grammar has the same number of a's as b's and, thus, is in L_3. We argue below that every string in L_3 can be derived using this grammar. Let w be an arbitrary string in L_3.

- If $w = \varepsilon$, then we use the rule $S \to \varepsilon$ to derive w.
- If $w = avb$ for some string v, then v must be in L_3. Since $v \in L_3$ and v is shorter than w, we have, by induction:
$$S \stackrel{*}{\Rightarrow} v$$
Using this, we have
$$S \Rightarrow aSbS \Rightarrow aSb \stackrel{*}{\Rightarrow} avb = w$$
- $w = bva$ for some $v \in L_3$. This is symmetric to the previous case and we have
$$S \Rightarrow bSaS \Rightarrow bSa \stackrel{*}{\Rightarrow} bva = w$$

[1]Every assignment must have at least one tricky question!

- $w = ava$, for some string v. Let n be the length of w. The number of a's in v is equal to $n/2 - 2$, and the number of b's in v is equal to $n/2$.

  Let $d(i)$ be the difference of the number of a's and the number of b's in the first i symbols of i. We note that

  $$d(1) = 1,$$

  $$d(n-1) = -1,$$

  $$\text{for all } i,\ d(i+1) \in \{d(i) - 1, d(i) + 1\}.$$

  Therefore, there is an index i such that

  $$d(i-1) = 1, d(i) = 0, d(i+1) = -1.$$

  We can write

  $$w = aubbu'a,$$

  where aub has length i, $bu'a$ has length $n - i$, u is in L_3 and $bu'a$ is in L_3. Since both u and $bu'a$ are shorter than w, we have, by induction,

  $$S \stackrel{*}{\Rightarrow} u$$

  and

  $$S \stackrel{*}{\Rightarrow} bu'a$$

  We conclude that

  $$S \Rightarrow aSbS \stackrel{*}{\Rightarrow} aubS \stackrel{*}{\Rightarrow} aubbu'a = w$$

- $w = bvb$, for some string v. This case is symmetric to the previous case.

Question 4: Give (deterministic or nondeterministic) pushdown automata that accept the following languages. For each pushdown automaton, start by explaining the algorithm in plain English, then mention the states that you are going to use, then explain the meaning of these states, and finally give the list of instructions.
(4.1) $\{a^n : n \geq 0\} \cup \{a^n b^n : n \geq 0\}$.
(4.2) $\{a^m b^n : m \geq 0, n \geq 0, m \neq n\}$.

Solution: For the language

$$\{a^n : n \geq 0\} \cup \{a^n b^n : n \geq 0\}$$

we can use a deterministic pushdown automaton. The approach is as follows:

- Walk along the input string from left to right.
- While reading a's: For each a, push a symbol S onto the stack.
- When we read the first non-a:

  - If it is $\square$: Do not move on the tape, empty the stack.
  - If it is b: For each b, pop one symbol from the stack.

- Tape alphabet $\Sigma = \{a, b\}$.
- Stack alphabet $\Gamma = \{\$, S\}$.

We use three states:

- q_a: This is the start state. If we are in this state, then we are reading the block of a's. The stack contains \$ at the bottom; the number of S-symbols on the stack is equal to the number of a's read so far.
- $q_\square$: The input string is of the form a^n, for some $n \geq 1$. We are at the first $\square$ after the input string.
- q_b: So far, we have read $a^n b^m$, for some $n \geq 1$ and $m \geq 1$.

The instructions are as follows.

- $q_a a\$ \to q_a R\$S$ (push S)
- $q_a aS \to q_a RSS$ (push S)
- $q_a \square\$ \to q_a N\varepsilon$ (input is empty, accept)
- $q_a \square S \to q_\square N\varepsilon$ (pop)
- $q_a b\$ \to q_a N\$$ (string starts with b, loop forever)
- $q_a bS \to q_b R\varepsilon$ (pop)
- $q_\square \square S \to q_\square N\varepsilon$ (pop)
- $q_\square \square\$ \to q_\square N\varepsilon$ (accept)
- $q_b a\$ \to q_b N\$$ (a to the right of b, loop forever)
- $q_b aS \to q_b NS$ (a to the right of b, loop forever)
- $q_b b\$ \to q_b N\$$ (too many b's, loop forever)
- $q_b bS \to q_b\varepsilon$ (pop)
- $q_b \square\$ \to q_b N\varepsilon$ (accept)
- $q_b \square S \to q_b NS$ (too many a's, loop forever)

For the language

$$\{a^m b^n : m \geq 0, n \geq 0, m \neq n\}$$

we can also use a deterministic pushdown automaton. The approach is as follows:

- Assume that the input string is of the form $a^m b^n$, for some $m \geq 1$ and $n \geq 1$.
- While reading a's: For each a, push a symbol S onto the stack.
- After we have read the first b: For each b,
  - if the top of the stack has S, then pop and go right.
  - if the top of the stack has \$, then we move right and do not change the stack.
- After we have read $\Box$:
  - if the top of the stack has \$, then loop forever.
  - if the top of the stack has S, then empty the stack and accept.
- We also have to handle the cases when the input string is empty, or of the form a^m for some $m \geq 1$, or of the form b^n for some $n \geq 1$.
- Tape alphabet $\Sigma = \{a, b\}$.
- Stack alphabet $\Gamma = \{\$, S\}$.

We use four states:

- q_a: This is the start state. If we are in this state, then we have read a^m for some $m \geq 0$. The stack contains \$ at the bottom; the number of S-symbols on the stack is equal to m.
- q_b: If we are in this state, then we have read $a^m b^n$ for some $m \geq 0$, $n \geq 1$, $m \geq n$. The stack contains \$ at the bottom; the number of S-symbols on the stack is equal to $m - n$.
- q_f: If we are in this state, then we have read the entire input string; it is of the form $a^m b^n$ for some $m > n \geq 0$. The head is at the first $\Box$ to the right of the input. We are going to empty the stack and accept the input.
- q'_f: If we are in this state, then we have read $a^m b^n$ for some $n > m \geq 0$. The stack contains only \$. We keep on reading: For each b, move right and do not change the stack. If we read a, loop forever. If we read $\Box$, accept the input string.

The instructions are as follows.

- $q_a \Box \$ \to q_a N \$$ (input is empty, loop forever)

- $q_a \square S \to q_f N \varepsilon$ (input is a^m for some $m \geq 1$; pop)
- $q_a a \$ \to q_a R \$ S$ (push S)
- $q_a a S \to q_a R S S$ (push S)
- $q_a b \$ \to q'_f R \$$ (string starts with b)
- $q_a b S \to q_b R \varepsilon$ (pop)
- $q_f \square S \to q_f N \varepsilon$ (pop)
- $q_f \square \$ \to q_f N \varepsilon$ (empty stack and accept)
- $q_b \square \$ \to q_b N \$$ (input is $a^m b^m$, loop forever)
- $q_b \square S \to q_f N \varepsilon$
- $q_b a \$ \to q_b N \$$ (a right of b, loop forever)
- $q_b a S \to q_b N S$ (a right of b, loop forever)
- $q_b b \$ \to q'_f R \$$
- $q_b b S \to q_b R \varepsilon$ (pop)
- $q'_f a \$ \to q'_f N \$$ (a right of b, loop forever)
- $q'_f b \$ \to q'_f R \$$
- $q'_f \square \$ \to q'_f N \varepsilon$ (empty stack and accept)

Question 5: Prove that the following languages are not context-free:
(5.1) $\{a^m b^n a^m b^n : m \geq 0, n \geq 0\}$.
(5.2) $\{a^m b^n c^k : 1 \leq m < n < k < 2m\}$.

Solution: First, we prove that the language

$$A = \{a^m b^n a^m b^n : m \geq 0, n \geq 0\}$$

is not context-free.

Assume that A is context-free. By the Pumping Lemma, there is an integer $p \geq 1$, such that for all strings $s \in A$ with $|s| \geq p$, the following holds: We can write $s = uvxyz$, where

1. vy is non-empty,
2. vxy has length at most p,
3. the string $uv^i xy^i z$ is in A, for all $i \geq 0$.

Consider the pumping length p. We choose $s = a^p b^p a^p b^p$. Then s is a string in A, and the length of s is $4p$, which is at least p (because $p \geq 1$). Thus, we can write $s = uvxyz$ such that 1., 2., and 3. above hold.

Case 1: vxy is in the left half of s. Let $s' = uvvxyyz$. The left part of s' has more than p many a's or more than p many $b's$. However, the right part of s' has p many a's and p many $b's$. Thus, s' is not in A. However, by the Pumping Lemma, s' is in A. This is a contradiction.

Case 2: vxy is in the $b^p a^p$-part of s. By the same reasoning as above, s' is not in A. However, by the Pumping Lemma, s' is in A. This is a contradiction.

Case 3: vxy is in the right half of s. By the same reasoning as above, s' is not in A. However, by the Pumping Lemma, s' is in A. This is a contradiction.

Since $|vxy| \leq p$, we have covered all possible cases. In each case, we arrived at a contradiction. We conclude that A is not context-free.

Next, we prove that the language

$$B = \{a^m b^n c^k : 1 \leq m < n < k < 2m\}$$

is not context-free.

Assume that B is context-free. By the Pumping Lemma, there is an integer $p \geq 1$, such that for all strings $s \in B$ with $|s| \geq p$, the following holds: We can write $s = uvxyz$, where

1. vy is non-empty,
2. vxy has length at most p,
3. the string $uv^i xy^i z$ is in B, for all $i \geq 0$.

Remark: We may assume that $p \geq 3$: If $p = 1$ or $p = 2$, then we can replace p by 3. Since the Pumping Lemma holds for the old p, it also holds for the new p.

Consider the pumping length $p \geq 3$. We choose the string

$$s = a^p b^{p+1} c^{p+2}.$$

Since $p \geq 3$, the string s is in B. Moreover, $|s| = 3p + 3 \geq p$. Thus, we can write $s = uvxyz$ such that 1., 2., and 3. above hold.

Case 1: vxy does not contain c. Let $s' = uvvxyyz$. In s', the number of c's is equal to $p+2$. Moreover, the number of a's is at least $p + 1$ or the number of b's is at least $p + 2$.

If, in s', the number of b's is at least $p + 2$, then the number of b's is at least the number of c's. Thus, s' is not in B. This is a contradiction.

If, in s', the number of b's is equal to $p + 1$, then the number of a's is at least $p + 1$. Therefore, in s', the number of a's is at least the number of b's. Thus, s' is not in B. This is a contradiction.

Case 2: vxy does not contain a. Let $s' = uxz$. In s', the number of a's is equal to p. Moreover, the number b's is at most p or the number of $c's$ is at most $p+1$.

If, in s', the number of b's is at most p, then the number of $a's$ is at least the number of b's. Thus, s' is not in B. This is a contradiction.

If, in s', the number of b's is at least $p+1$, then the number of c's is at most $p+1$. Thus, the number of b's is at least the number c's. Thus, s' is not in B. This is a contradiction.

Since $|vxy| \leq p$, we have covered all possible cases. In each case, we arrived at a contradiction. We conclude that B is not context-free.

Question 6:
(6.1) Let $L = \{a^i b^{2i} c^j : i \geq 0, j \geq 0\}$. Prove that L is a context-free language.
(6.2) Let $L' = \{a^j b^i c^{2i} : i \geq 0, j \geq 0\}$. Prove that L' is a context-free language.
(6.3) Prove that the language $L \cap L'$ is not context-free.

Solution: First we show that $L = \{a^i b^{2i} c^j : i \geq 0, j \geq 0\}$ is context-free. Consider the following context-free grammar:

$$\begin{array}{lcl} S & \rightarrow & AC \\ A & \rightarrow & \varepsilon \mid aAbb \\ C & \rightarrow & \varepsilon \mid Cc \end{array}$$

From A, we can derive all strings $a^i b^{2i}$ for $i \geq 0$. From C, we can derive all strings c^j for $j \geq 0$. Thus, from S, we can derive all strings in L. It is clear that no other strings can be derived from S. This shows that L is context-free.

Next we show that $L' = \{a^j b^i c^{2i} : i \geq 0, j \geq 0\}$ is context-free. Consider the following context-free grammar:

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & \varepsilon \mid Aa \\ B & \rightarrow & \varepsilon \mid bBcc \end{array}$$

From A, we can derive all strings a^j for $j \geq 0$. From B, we can derive all strings $b^i c^{2i}$ for $i \geq 0$. Thus, from S, we can derive all strings in L'. It is clear that no other strings can be derived from S. This shows that L' is context-free.

Finally, we show that $L \cap L'$ is not context-free. Any string in L is of the form $a^i b^{2i} c^j$, whereas any string in L' is of the form $a^\ell b^k c^{2k}$. For such strings to be in $L \cap L'$, we need $i = \ell$, $2i = k$, and $j = 2k$. Thus,

$$L \cap L' = \{a^\ell b^{2\ell} c^{4\ell} : \ell \geq 0\}.$$

Assume that $L \cap L'$ is context-free. By the Pumping Lemma, there is an integer $p \geq 1$, such that for all strings $s \in L \cap L'$ with $|s| \geq p$, the following holds: We can write $s = uvxyz$, where

1. vy is non-empty,

2. vxy has length at most p,

3. the string uv^ixy^iz is in $L \cap L'$, for all $i \geq 0$.

Consider the pumping length p. We choose $s = a^p b^{2p} c^{4p}$. Then s is a string in $L \cap L'$, and the length of s is $7p$, which is at least p. Thus, we can write $s = uvxyz$ such that 1., 2., and 3. above hold.

Case 1: vxy does not contain c. Consider $s' = uvvxyyz$. In s', the number of c's is equal to $4p$. Moreover, the number of a's is at least $p + 1$ or the number of b's is at least $2p + 1$. Therefore, s' is not in $L \cap L'$. This is a contradiction.

Case 2: vxy does not contain a. Consider $s' = uvvxyyz$. In s', the number of a's is equal to p. Moreover, the number of b's is at least $2p + 1$ or the number of c's is at least $4p + 1$. Therefore, s' is not in $L \cap L'$. This is a contradiction.

Since $|vxy| \leq p$, we have covered all cases. In each case, we arrived at a contradiction. We conclude that $L \cap L'$ is not context-free.