COMP 3803 — Solutions Assignment 3

Question 1: Write your name and student number.

Solution: Alphonso Davies, 19

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

$$S \rightarrow \varepsilon \mid aSbS \mid bSaS$$

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: Each time we apply one of the rules $S \rightarrow aSbS$ or $S \rightarrow bSaS$, we add one a and one b. Based on this, it looks like $L(G) = L$, where

$$L = \{w \in \{a, b\}^* : \text{ the number of } a\text{'s in } w \text{ equals the number of } b\text{'s in } w\}.$$

To prove that this is correct, we have to prove that $L(G) \subseteq L$ and $L \subseteq L(G)$.

Proof that $L(G) \subseteq L$: We will prove that every string w in $L(G)$ is also in L. The proof is by induction on the length of w. If $w = \varepsilon$, then $w \in L(G)$, because $S \rightarrow \varepsilon$ is a rule. In this case, the number of a's in w is equal to the number of b's in w, because both numbers are zero. Thus, $w \in L$.

Assume that $w \neq \varepsilon$. Since $w \in L(G)$, we can derive w from the start variable S. The first rule to apply is either $S \rightarrow aSbS$ or $S \rightarrow bSaS$. When we apply either of these, we add one a and one b. In the rest of the derivation, we apply rules to the two occurences of S. Each of these occurrences leads to a substring of w; since it is shorter than w, we apply induction and conclude that in this substring, the number of a's is equal to the number of b's. It follows that in the entire string w, the number of a's is equal to the number of b's. Thus, $w \in L$.

Proof that $L \subseteq L(G)$: We will prove that every string w in L is also in $L(G)$. The proof is by induction on the length of w. If $w = \varepsilon$, then $w \in L(G)$, because $S \rightarrow \varepsilon$ is a rule.

Assume that $w = w_1 w_2 \dots w_n$, where $n \geq 2$. By symmetry, we may assume that $w_1 = a$. Let $m \geq 2$ be the smallest integer such that in the string $w_1 w_2 \dots w_m$, the number of a's is equal to the number of b's. Note that m exists, because this property holds for $m = n$.

- By definition of m, w_m must be b.
- In the string $w_2 \dots w_{m-1}$, the number of a's is equal to the number of b's. Thus, by induction, we can derive this string from the start variable S.

- In the string $w_{m+1} \dots w_n$, the number of a's is equal to the number of b's. Thus, by induction, we can derive this string from the start variable S.
- To derive w, we first apply the rule $S \to aSbS$. Then we apply rules to derive $w_2 \dots w_{m-1}$ from the leftmost S. Finally, we apply rules to derive $w_{m+1} \dots w_n$ from the rightmost S. Overall, this shows that we can derive w from S.

Question 3: Give context-free grammars that generate the following languages. For each case, justify your answer.
(3.1) The language of all strings over the alphabet $\{a, b\}$ that have an even number of b's. Note that zero is an even number and ε is in this language.
(3.2) $\{ba^n ba^n b : n \geq 0\}$.

Solution: The first language is described by the regular expression

$$(a^* ba^* ba^*)^* .$$

This leads to the context-free grammar $G = (V, \Sigma, R, S)$, where $V = \{S, A\}$, $\Sigma = \{a, b\}$, and R consists of the rules

$$\begin{array}{lcl} S & \to & \varepsilon \mid AbAbAS \\ A & \to & \epsilon \mid aA \end{array}$$

From A, we can derive all strings of the form a^n with $n \geq 0$. Thus, from S, we can derive all strings of the form

$$a^n ba^m ba^k S$$

with $n \geq 0$, $m \geq 0$, $k \geq 0$. Note that in such a derivation, we add exactly two b's to the string. By repeating, this, we get all strings with an even number of b's (and nothing else).

Now we look at the second language. Any string $ba^n ba^n b$ in this language is either

- bbb (in case $n = 0$) or
- ba, followed by $a^{n-1} ba^{n-1}$, followed by ab (in case $n \geq 1$).

This leads to the context-free grammar $G = (V, \Sigma, R, S)$, where $V = \{S, X\}$, $\Sigma = \{a, b\}$, and R consists of the rules

$$\begin{array}{lcl} S & \to & bbb \mid baXab \\ X & \to & b \mid aXa \end{array}$$

From X, we can derive all strings of the form $a^{n-1} ba^{n-1}$ with $n \geq 1$. From S, we can derive bbb and all strings of the form

$$ba \left(a^{n-1} ba^{n-1}\right) ab = ba^n ba^n b.$$

Nothing else can be derived from S.

Question 4: A regular expression over the alphabet $\{a, b\}$ can be described by a string whose symbols belong to the set

$$\Sigma = \{\emptyset, \varepsilon, a, b, \cup, ^*, (,)\}.$$

For example, the following string over Σ is a regular expression:

$$((a \cup b)^* ab)^* \cup \emptyset \cup \varepsilon.$$

Give a context-free grammar that generates all regular expressions over the alphabet $\{a, b\}$. The set of terminals of the grammar is Σ.

Solution: All regular expressions are obtained by the following "rules":

1. $\emptyset$,
2. ε,
3. a,
4. b,
5. union of two regular expressions,
6. concatenation of two regular expressions,
7. star of a regular expression,
8. add brackets in the appropriate places.

Based on this, we get the context-free grammar that has the following rules:

$$S \to \emptyset \mid \varepsilon \mid a \mid b \mid S \cup S \mid SS \mid S^* \mid (S).$$

For example, the regular expression $(a \cup b)^*$ is obtained as follows:

$$\begin{aligned} S &\Rightarrow S^* \\ &\Rightarrow (S)^* \\ &\Rightarrow (S \cup S)^* \\ &\Rightarrow (a \cup S)^* \\ &\Rightarrow (a \cup b)^* \end{aligned}$$

Question 5: 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.
(5.1) $\{a^n b a^n : n \geq 0\}$.

(5.2) $\{a^m b^n : 0 \leq n \leq m \leq 2n\}$.

Solution: For the language

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

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

- Walk along the input string from left to right.
- While reading the leftmost a-block: For each a, push a symbol S onto the stack. Stay in the start state.
- At the first b, switch to a new state and move one cell to the right.
- Continue in this new state. For each a, pop the top symbol S from the stack.
- Tape alphabet $\Sigma = \{a, b\}$.
- Stack alphabet $\Gamma = \{\$, S\}$.

We use two states:

- q_0: This is the start state. If we are in this state, then we are reading the leftmost 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_1: We have seen the first b and are currently reading the rightmost 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 in the leftmost a-block minus the number a's in the rightmost a-block that we have seen so far.

The instructions are as follows.

- $q_0 a\$ \to q_0 R\$S$ (push S)
- $q_0 aS \to q_0 RSS$ (push S)
- $q_0 b\$ \to q_1 R\$$ (first b)
- $q_0 bS \to q_1 RS$ (first b)
- $q_0 \square\$ \to q_0 N\$$ (no b in string, loop forever)
- $q_0 \square S \to q_0 NS$ (no b in string, loop forever)
- $q_1 a\$ \to q_1 N\$$ (more a's on the right than on the left; loop forever)
- $q_1 aS \to q_1 R\varepsilon$ (pop)

- $q_1 b\$ \to q_1 N\$$ (string has at least two b's; loop forever)
- $q_1 bS \to q_1 NS$ (string has at least two b's; loop forever)
- $q_1 \square\$ \to q_1 N\varepsilon$ (accept)
- $q_1 \square S \to q_1 NS$ (more a's on the left than on the right; loop forever)

For the language

$$\{a^m b^n : 0 \leq n \leq m \leq 2n\}$$

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

- Walk along the input string from left to right.
- While reading the a-block: For each a, push a symbol S onto the stack.
- While reading the b-block: For each b, use nondeterminism to pop one or two symbols.
- For example, if the input string is $a^5 b^3$: After having read the a-block, the stack has five symbols S. For the first b, we do two pops, for the second b, we do one pop, for the third b, we do two pops.
- Tape alphabet $\Sigma = \{a, b\}$.
- Stack alphabet $\Gamma = \{\$, S\}$.

We will use three states:

- q_0: This is the start state. If we are in this state, then we are reading the a-block.
- q_1: If we are in this state, then we are reading the b-block. If we read b, then we either do one pop and stay in q_1, or we do one pop and switch to state q_1'.
- q_1': If we are in this state, then we are reading the b-block. If we read b, then we do one pop and go back to q_1.

The instructions are as follows.

- $q_0 a\$ \to q_0 R\$S$ (push S)
- $q_0 aS \to q_0 RSS$ (push S)
- $q_0 b\$ \to q_0 N\$$ ($m = 0$, $n \geq 1$, loop forever)
- $q_0 bS \to q_1 RS$ (read the first b)
- $q_0 \square\$ \to q_0 N\varepsilon$ ($m = 0$, $n = 0$, accept)

- $q_0 \square S \to q_0 N S$ ($m \geq 1$, $n = 0$, loop forever)
- $q_1 a \$ \to q_1 N \$$ (a to the right of b, loop forever)
- $q_1 a S \to q_1 N S$ (a to the right of b, loop forever)
- $q_1 b \$ \to q_1 N \$$ (too many b's, loop forever)
- $q_1 b S \to q_1 R \varepsilon$ (one pop)
- $q_1 b S \to q_1' N \varepsilon$ (one pop, second pop comes later)
- $q_1 \square \$ \to q_1 N \varepsilon$ (accept)
- $q_1 \square S \to q_1 N S$ (too many a's, loop forever)
- $q_1' b \$ \to q_1' N \$$ (too many b's, loop forever)
- $q_1' b S \to q_1 R \varepsilon$ (second pop)
- Note: If we are in state q_1', then we must read b.

Question 6: Let L_1 and L_2 be two context-free languages over the same alphabet Σ. Assume that you are given a (deterministic or nondeterministic) pushdown automaton M_1 that accepts L_1, and you are also given a (deterministic or nondeterministic) pushdown automaton M_2 that accepts L_2.

Explain how you can combine M_1 and M_2 into one single (deterministic or nondeterministic) pushdown automaton that accepts the concatenation $L_1 L_2$.

Your explanation may be in plain English and may use figures and diagrams.

Solution: Let us first recall when the pushdown automaton (PDA) M_1 accepts an input string v:

- At the start, the tape head is at the leftmost symbol of v and the stack only contains the special symbol \$.
- The computation stops as soon as the stack is empty; the string v is accepted if, at that moment, the tape head is at the $\square$ symbol immediately to the right of v.
- When the tape head reaches this $\square$ symbol, it may happen that the stack contains symbols other than \$.
  - For example, assume that $L_1 = \{a^m b^n : m > n \geq 1\}$. The obvious PDA pushes one symbol S for each a; then it does one pop for each b. When the head reaches the first $\square$, the stack contains $m - n$ symbols S. The PDA then pops all of them and, finally, pops \$.

Thus, in general, when the PDA reaches $\square$, it "fools around" with the stack and, at the end, pops \$.

The question asks for a PDA that accepts $L_1 L_2$. Any string u in this language is of the form $u = vw$ with $v \in L_1$ and $w \in L_2$. Thus, the PDA M_1 must accept v and the PDA M_2 must accept w.

- Let us denote the target PDA by M. The input to M is a string u.
- M will run M_1 on input u. It will use nondeterminism to guess when it has finished reading v and knows that $v \in L_1$. Then, it will run M_2 on the rest of the input string to check that $w \in L_2$.
- How does M know that it has finished reading v and that $v \in L_1$? Note that M_1 knows this if it reads $\square$ and the stack only has \$; at that moment, M_1 empties the stack. However, M will not read $\square$, because it enters w (unless w is empty). To solve this issue, we will introduce a new state and extra instructions; see below.
- Let q_1 be the start state of M_1 and let q_2 be the start state of M_2. The PDA M will have q_1 as its start state. Initially, M will run M_1. We introduce a new state q_3; its meaning is "M has finished reading v, it is at the leftmost symbol of w, and it is fooling around with the stack like M_1 does". When the stack only has \$, instead of popping this symbol (as M_1 would do), M will not change the stack and switch to the start state q_2 of M_2. Then, M will run M_2.
- Based on this, we get the following NPDA M:
  - The states of M are all states of M_1, all states of M_2, and the new state q_3.
  - All instructions of M_1 are also instructions of M.
  - All instructions of M_2 are also instructions of M.
  - **Warning:** Below, a few edge cases may be missing.
  - We add the following instructions; these are used to simulate that M switches from M_1 to M_2:
    - For each instruction of M_1 of the form

      $$q\square S \to q'Nx,$$

      with $S \neq \$$ or $x \neq \varepsilon$, we add the new instructions

      $$qaS \to q_3 Nx$$

      for all symbols $a \in \Sigma$.

* For each instruction of M_1 of the form

$$q\square\$ \to q'N\varepsilon,$$

  we add the new instructions

$$qa\$ \to q_2N\$$$

  for all symbols $a \in \Sigma$.
* For each $S \neq \$$ and each symbol $a \in \Sigma$, we add the instruction

$$q_3aS \to q_3N\varepsilon$$

* Finally, for each symbol $a \in \Sigma$, we add the instruction

$$q_3a\$ \to q_2N\$$$

Question 7: Prove that the following languages are not context-free. In both cases, the alphabet is $\{a, b, c\}$.
(7.1) $\{a^kb^mc^n : 0 \leq k < m < n\}$.
(7.2) $\{a^kb^mc^n : k \geq 0, m \geq 0, n \geq 0, k^2 + m^2 = n^2\}$. *Hint:* $3^2 + 4^2 = 5^2$.

Solution: First, we prove that the language

$$A = \{a^kb^mc^n : 0 \leq k < m < n\}.$$

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^ixy^iz is in A, for all $i \geq 0$.

Consider the pumping length p. We choose $s = a^pb^{p+1}c^{p+2}$. This string is in A, and its length is $3p + 3$, 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: vy is within the a-block.

Consider the string uv^2xy^2z. The number of a's is at least $p + 1$, whereas the number of b's is equal to $p + 1$. Thus, this string is not in A. This is a contradiction.

Case 2: vy contains at least one b, but no c.

Consider the string uv^2xy^2z. The number of b's is at least $p + 2$, whereas the number of c's is equal to $p + 2$. Thus, this string is not in A. This is a contradiction.

Case 3: vy is within the c-block.

Consider the string uv^0xy^0z. The number of b's is equal to $p+1$, whereas the number of c's is at most $p+1$. Thus, this string is not in A. This is a contradiction.

Case 4: vy contains at least one b and at least one c.

Consider the string uv^0xy^0z. The number of a's is equal to p, whereas the number of b's is at most p. Thus, this string is not in A. This is a contradiction.

Since $|vxy| \leq p$, we have covered all cases. In each of the four cases, we have obtained a contradiction. Therefore, A is not a context-free language.

Next we prove that the language

$$B = \{a^k b^m c^n : k \geq 0, m \geq 0, n \geq 0, k^2 + m^2 = n^2\}$$

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^ixy^iz is in B, for all $i \geq 0$.

Consider the pumping length p. We choose $s = a^{3p}b^{4p}c^{5p}$. Since $(3p)^2 + (4p)^2 = (5p)^2$, the string s is in B, and its length is $12p$, 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: vy is within the ab-block.

Consider the string uv^2xy^2z. The number of a's is larger than $3p$ or the number of b's is larger than $4p$ (or both). The number of c's is equal to $5p$. Thus, the square of the number of a's plus the square of the number of b's is larger than the square of the number of c's. Therefore, uv^2xy^2z is not in B. This is a contradiction.

Case 2: vy is within the c-block.

Consider the string uv^2xy^2z. The number of a's is equal to $3p$ and the number of b's is equal to $4p$. The number of c's is larger than $5p$. Thus, the square of the number of a's plus the square of the number of b's is less than the square of the number of c's. Therefore, uv^2xy^2z is not in B. This is a contradiction.

Case 3: vy contains at last one b and at least one c.

We write $vy = b^jc^k$, where $j \geq 1$ and $k \geq 1$.

If we pump down, we obtain the string

$$uv^0xy^0z = a^{3p}b^{4p-j}c^{5p-k}.$$

Since this string is in B, we have

$$(3p)^2 + (4p-j)^2 = (5p-k)^2,$$

which is equivalent to

$$-8pj + j^2 = -10pk + k^2. \tag{1}$$

If we pump up , we obtain the string

$$uv^2xy^2z = a^{3p}b^{4p+j}c^{5p+k}.$$

Since this string is in B, we have

$$(3p)^2 + (4p + j)^2 = (5p + k)^2,$$

which is equivalent to

$$8pj + j^2 = 10pk + k^2. \tag{2}$$

By adding (1) and (2), we get

$$2j^2 = 2k^2,$$

which is equivalent to

$$j = k.$$

By setting $j = k$ in (2), we get

$$8pj + j^2 = 10pj + j^2,$$

which is equivalent to

$$pj = 0.$$

This is a contradiction, because $p \geq 1$ and $j \geq 1$.

Case 4: vy contains at last one a and at least one c.

In this case, $|vxy| \geq 4p + 2 > p$, which is a contradiction.

Conclusion: In each of the four cases, we have obtained a contradiction. Therefore, B is not a context-free language.