COMP2270/6270 – Theory of Computation
Eighth Week

School of Electrical Engineering & Computing
The University of Newcastle

Exercise 1) For each of the following languages L, state whether L is regular, context-free but not regular, or not context-free and prove your answer.

a) $L = \{xy : x, y \in \{\mathtt{a}, \mathtt{b}\}^* \text{ and } |x| = |y|\}$.

Regular. $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : |w| \text{ is even}\}$
$= (\mathtt{aa} \cup \mathtt{ab} \cup \mathtt{ba} \cup \mathtt{bb})^*$

b) $L = \{(\mathtt{ab})^n \mathtt{a}^n \mathtt{b}^n : n > 0\}$.

Not context-free. We prove this with the Pumping Theorem. Let $w = (\mathtt{ab})^k \mathtt{a}^k \mathtt{b}^k$. Divide w into three regions: the ab region, the a region, and the b region. If either v or y crosses the boundary between regions 2 and 3 then pump in once. The resulting string will have characters out of order. We consider the remaining alternatives for where nonempty v and y can occur:
(1, 1) If $|vy|$ is odd, pump in once and the resulting string will have characters out of order. If it is even, pump in once. The number of ab's will no longer match the number of a's in region 2 or b's in region 3.
(2, 2) Pump in once. More a's in region 2 than b's in region 3.
(3, 3) Pump in once. More b's in region 3 than a's in region 2.
v or y crosses the boundary between 1 and 2: Pump in once. Even if v and y are arranged such that the characters are not out of order, there will be more ab pairs than there are b's in region 3.
(1, 3) $|vxy|$ must be less than or equal to k.

c) $L = \{\mathtt{a}^i \mathtt{b}^n : i, n > 0 \text{ and } i = n \text{ or } i = 2n\}$.

Context-free, not regular. L can be generated by the following context-free grammar:

$S \rightarrow A \mid B$
$A \rightarrow \mathtt{a}A\mathtt{b} \mid \mathtt{ab}$
$B \rightarrow \mathtt{aa}B\mathtt{b} \mid \mathtt{aab}$

L is not regular, which we show by pumping: Let $w = \mathtt{a}^k \mathtt{b}^k$. Pump out. To get a string in L, i must be equal to n or greater than n (in the case where $i = 2n$). Since we started with $i = n$ and then decreased i, the resulting string must not be in L.

d) $L = \{\mathtt{a}^i : i \geq 0\}\ \{\mathtt{b}^i : i \geq 0\}\ \{\mathtt{a}^i : i \geq 0\}$.

Regular. $L = \mathtt{a}^*\mathtt{b}^*\mathtt{a}^*$.

e) $L(G)$, where $G = S \rightarrow \mathtt{a}S\mathtt{a}$
$S \rightarrow SS$
$S \rightarrow \varepsilon$

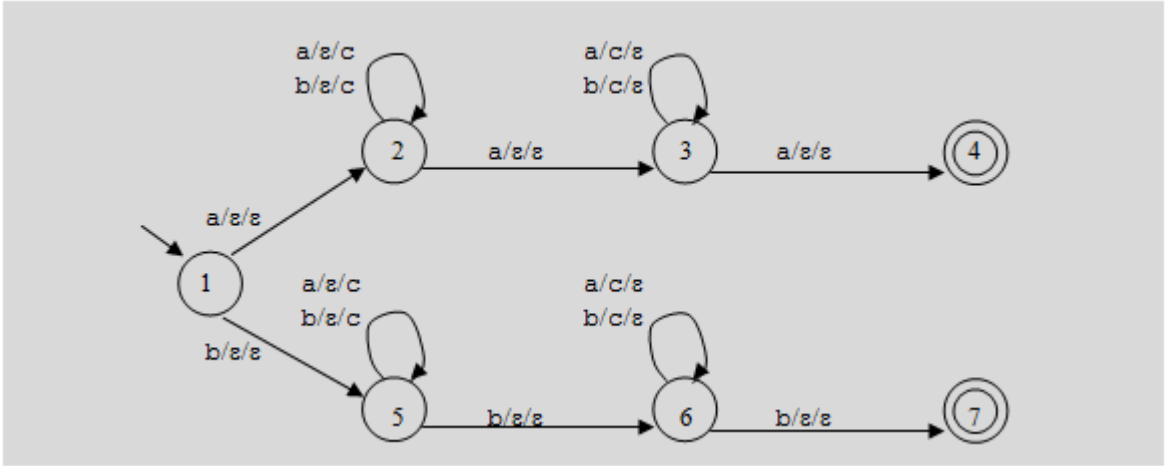
Regular. $L = (\mathtt{aa})^*$.

Exercise 2) Let $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* :$ the first, middle, and last characters of w are identical$\}$.

a) Show a context-free grammar for L.

$S \rightarrow A \mid B$
$A \rightarrow \mathtt{a}\, M_A\, \mathtt{a}$
$M_A \rightarrow CM_AC \mid \mathtt{a}$
$B \rightarrow \mathtt{b}\, M_B\, \mathtt{b}$
$M_B \rightarrow CM_BC \mid \mathtt{b}$
$C \rightarrow \mathtt{a} \mid \mathtt{b}$

b) Show a natural PDA that accepts L.



c) Prove that L is not regular.

If L is regular, then so is $L' = L \cap \mathtt{ab}^*\mathtt{ab}^*\mathtt{a}$. But we show that it is not. Let $w = \mathtt{ab}^k\mathtt{ab}^k\mathtt{a}$.
1| 2|3| 4|5

If y crosses numbered regions, pump in once. The resulting string will not be in L' because the letters will be out of order. If y is in region 1, 3, or 5, pump out and the resulting string will not be in L'. Because y must occur within the first k characters, the only place it can fall is within region 2. Pump in once. If $|y|$ is odd, the resulting string is not in L' because it has no middle character. If $|y|$ is even and thus at least 2, the resulting string is not in L' because its middle character is b, yet its first and last characters are a's.

Exercise 3) Without using the Pumping Theorem, prove that $L = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* : \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w) = \#_{\mathtt{c}}(w)$ and $\#_{\mathtt{a}}(w) > 50\}$ is not context-free.

Let $L_1 = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* : \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w) = \#_{\mathtt{c}}(w)$ and $\#_{\mathtt{a}}(w) \leq 50\}$. L_1 is regular, and thus also context-free, because it is finite. Suppose L were context-free. Then $L_2 = L \cup L_1$ would also be context-free, since the context-free languages are closed under union. But $L_2 = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* : \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w) = \#_{\mathtt{c}}(w)\}$,which we have shown is not context-free. [EXAMPLE 13.8: page 294]

Exercise 4) Give an example of a context-free language L ($\neq \Sigma^*$) that contains a subset L_1 that is not context-free. Prove that L is context free. Describe L_1 and prove that it is not context-free.

Let $L = \{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^p : n \geq 0, m \geq 0, p \geq 0,$ and $n = m$ or $m = p\}$. L is context free because we can build a nondeterministic PDA M to accept it. M has two forks, one of which compares n to m and the other of

which compares m to p (skipping over the a's). $L_1 = \{a^n b^m c^p : n = m \text{ and } m = p\}$ is a subset of L. But $L_1 = A^n B^n C^n = \{a^n b^n c^n : n \geq 0\}$, which we have shown is not context free.

Exercise 5) Let $L_1 = L_2 \cap L_3$.

a) Show values for L_1, L_2, and L_3, such that L_1 is context-free but neither L_2 nor L_3 is.

Let: $L_1 = \{a^n b^n : n \geq 0\}$.
$L_2 = \{a^n b^n c^j : j \leq n\}$.
$L_3 = \{a^n b^n c^j : j = 0 \text{ or } j > n\}$.

b) Show values for L_1, L_2, and L_3, such that L_2 is context-free but neither L_1 nor L_3 is.

Let: $L_2 = a^*$.
$L_1 = \{a^n : n \text{ is prime}\}$.
$L_3 = \{a^n : n \text{ is prime}\}$.

Exercise 6) Give an example of a context-free language L, other than one of the ones in the book, where $\neg L$ is not context-free.

Let $L = \{a^i b^j c^k : i \neq j \text{ or } j \neq 2k\}$. L is context free. Another way to describe it is:

$L = \{a^i b^j c^k : i > j \text{ or } i < j \text{ or } j > 2k \text{ or } j < 2k\}$.

We can write a context-free grammar for L by writing a grammar for each of its sublanguages. So we have:

$S \rightarrow S_1 C \mid S_2 C \mid A S_3 \mid A S_4$
$S_1 \rightarrow a S_1 b \mid a S_1 \mid a$ /* More a's than b's.
$S_2 \rightarrow a S_2 b \mid S_2 b \mid b$ /* More b's than a's.
$S_3 \rightarrow bb S_3 c \mid b S_3 \mid b$ /* $j > 2k$.
$S_4 \rightarrow bb S_4 c \mid b S_4 c \mid S_4 c \mid c$ /* $j < 2k$.
$A \rightarrow aA \mid \varepsilon$
$C \rightarrow cC \mid \varepsilon$

$\neg L = \{a^i b^j c^k : i = j \text{ and } j = 2k\} \cup \{w \in \{a, b, c\}^* : \text{the letters are out of order}\}$. $\neg L$ is not context-free. If it were, then $L' = \neg L \cap a^* b^* c^*$ would also be context-free. But $L' = \{a^i b^j c^k : i = j \text{ and } j = 2k\}$, which can easily be shown not be be context-free by using the Pumping Theorem, letting $w = a^k b^k c^{2k}$.

Exercise 7) Theorem 13.7 tells us that the context-free languages are closed under intersection with the regular languages. Prove that the context-free languages are also closed under union with the regular languages.

Every regular language is also context-free and the context-free languages are closed under union.

Extra from book:

[Chapter 13 Exercise 1] For each of the following languages L, state whether L is regular, context-free but not regular, or not context-free and prove your answer.

a) $\{x\#y : x, y \in \{0, 1\}^* \text{ and } x \neq y\}$.

Context-free not regular. We can build a PDA M to accept L. All M has to do is to find one way in which x and y differ. We sketch its construction: M starts by pushing a bottom of stack marker Z onto

the stack. Then it nondeterministically chooses to go to state 1 or 2. From state 1, it pushes the characters of x, then starts popping the characters of y. It accepts if the two strings are of different lengths. From state 2, it must accept if two equal length strings have at least one different character. So M starts pushing a % for each character it sees. It nondeterministically chooses a character on which to stop. It remembers that character in its state (so it branches and there are two similar branches from here on). It reads the characters up to the # and does nothing with them. Starting with the first character after the #, it pops one % for each character it reads. When the stack is empty it checks to see whether the next input character matches the remembered character. If it does not, it accepts.

b) $\{wx : |w| = 2 \cdot |x|$ and $w \in \mathtt{a}^+\mathtt{b}^+$ and $x \in \mathtt{a}^+\mathtt{b}^+\}$.

Context-free, not regular. L can be accepted by a PDA M that pushes one character for each a and b in w and then pops two characters for each a and b in x.

L is not regular, which we show by pumping. Note that the boundary between the w region and the x region is fixed; it's immediately after the last b in the first group. Choose to pump the string $w = \mathtt{a}^{2k}\mathtt{b}^{2k}\mathtt{a}^k\mathtt{b}^k$. $y = \mathtt{a}^p$, for some nonzero p. Pump in or out. The length of w changes but the length of x does not. So the resulting string is not in L.

c) $\{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^k : n, m, k \geq 0$ and $m \leq min(n, k)\}$.

Not context-free. We prove it using the Pumping Theorem. Let k be the constant from the Pumping Theorem and let $w = \mathtt{a}^k\mathtt{b}^k\mathtt{c}^k$. Let region 1 contain all the a's, region 2 contain all the b's, and region 3 contain all the c's. If either v or y crosses numbered regions, pump in once. The resulting string will not be in L because it will violate the form constraint. We consider the remaining cases for where nonempty v and y can occur:

(1, 1): Pump out once. This reduces the number of a's and thus the *min* of the number of a's and c's. But the number of b's is unchanged so it is greater than that minimum.
(2, 2): Pump in once. The *min* of the number of a's and c's is unchanged. But the number of b's is increased and so it is greater than that minimum.
(3, 3): Same argument as (1, 1) but reduces the number of c's.
(1, 2), (2, 3): Pump in once. The *min* of the number of a's and c's is unchanged. But the number of b's is increased and so it is greater than that minimum.
(1, 3): Not possible since $|vxy|$ must be less than or equal to k.

d) $\{xyx^R : x \in \{\mathtt{0}, \mathtt{1}\}^+$ and $y \in \{\mathtt{0}, \mathtt{1}\}^*\}$.

Regular. There is no reason to let x be more than one character. So all that is required is that the string have at least two characters and the first and last must be the same. $L = (\mathtt{0}\ (\mathtt{0} \cup \mathtt{1})^*\ \mathtt{0}) \cup (\mathtt{1}\ (\mathtt{0} \cup \mathtt{1})^*\ \mathtt{1})$.

f) $\{xyx^R : x \in \{\mathtt{0}, \mathtt{1}\}^+$ and $y \in \{\mathtt{0}, \mathtt{1}\}^*\}$.

Regular. There is no reason to let x be more than one character. So all that is required is that the string have at least two characters and the first and last must be the same. $L = (\mathtt{0}\ (\mathtt{0} \cup \mathtt{1})^*\ \mathtt{0}) \cup (\mathtt{1}\ (\mathtt{0} \cup \mathtt{1})^*\ \mathtt{1})$.

g) $\{ww^Rw : w \in \{\mathtt{a}, \mathtt{b}\}^*\}$.

Not context free. We prove it using the Pumping Theorem. Let $w = \mathtt{a}^k\mathtt{b}^k\mathtt{b}^k\mathtt{a}^k\mathtt{a}^k\mathtt{b}^k$.
1 | 2 | 3 | 4
In each of these cases, pump in once:

- If any part of v is in region 1, then to produce a string in L we must also pump a's into region 3. But we cannot since $|vxy| \leq k$.
- If any part of v is in region 2, then to produce a string in L we must also pump b's into region 4. But we cannot since $|vxy| \leq k$.

- If any part of *v* is in region 3, then to produce a string in *L* we must also pump a's into region 1. But we cannot since *y* must come after *v*.
- If any part of *v* is in region 4, then to produce a string in *L* we must also pump b's into region 2. But we cannot since y must come after *v*.

h) $\neg L_0$, where $L_0 = \{ww : w \in \{a, b\}^*\}$.

Context-free, not regular. Observe that *L* includes all strings in {a, b}* of odd length. It also includes all even length strings of the form *x*a*y z*b*q*, where |*x*| = |*z*| and |*y*| = |*q*|. In other words there is some position in the first half that contains an a, while the corresponding position in the second half contains b, or, similarly, where there's a b in the first half and an a in the second half. Those strings look like *x*b*y z*a*q*. The only thing that matters about *yz* is its length and we observe that |*yz*| = |*zy*|. So an alternative way to describe the strings of the form *x*a*y z*b*q* is *x*a*z y*b*q*, where |*x*| = |*z*| and |*y*| = |*q*|. With this insight, we realize that *L* =

$\{w \in \{a, b\}^* : |w| \text{ is odd}\} \cup$ [1]

$\{w \in \{a, b\}^* : w = x\text{a}z\, y\text{b}q, \text{ where } |x| = |z| \text{ and } |y| = |q|\} \cup$ [2]

$\{w \in \{a, b\}^* : w = x\text{b}z\, y\text{a}q, \text{ where } |x| = |z| \text{ and } |y| = |q|\}$ [3]

Now we can build either a grammar or a PDA for each piece.

The PDAs for [2] and [3] work by pushing a symbol onto the stack for each symbol in *x*, then guessing at the symbol to remember. Then they pop a symbol for each input symbol until the stack is empty. Then they push one for every input symbol until they guess that they're at the mismatched symbol. After reading it, they pop one from the stack for every input symbol. If the stack and the input run out at the same time, the second guess chose the matching position. If the two characters that are checked are different, accept, else reject.

A grammar for [2] is: $S \rightarrow S_a S_b$, $S_a \rightarrow CS_aC$, $S_a \rightarrow a$, $S_b \rightarrow CS_bC$, $S_b \rightarrow b$, $C \rightarrow a$, $C \rightarrow b$. The grammar for [3] is analogous except that its *S* rule is $S \rightarrow S_b S_a$.