CITS2211 Discrete Structures
Week 11 Tutorial PDAs

Topics: The pumping lemma for regular languages, Context-free languages, Context-free grammars, Push-down automata

1. State the Pumping Lemma for Regular Languages. Write out your answer in a way that helps you to remember the lemma.

Solution:

Theorem: If $L$ is a regular language then

$\exists$ an integer $p$ called the **pumping length** of $L$ such that

$\forall$ words $w \in L$ where $|w| > p$

$\exists$ an expression $w = xyz$ where

(a) $\forall i \geq 0. \ xy^iz \in L$
(b) $|y| \geq 1$
(c) $|xy| \leq p$

Note the $i \geq 1$ version is OK too.

2. Use the Pumping Lemma for Regular Languages to prove that the language all binary strings that have equal numbers of 0s and 1s is *not* regular.

Solution:

Suppose $L$ is regular.

Let the pumping length be $p$.

Choose $w = 0^p1^p$. Clearly $w \in L$ because it has $p$ 0s and $p$ 1s.

For any adversary choice of $xyz = w$, both $x$ and $y$ can only contain 0s since $|xy| \leq p$ (constraint (c)).

Let $x = 0^m$, $y = 0^n$ for some $m + n \leq p$.

The pumping lemma states that $xy^iz \in L$ but $xy^iz \notin L$ because $xy^iz$ contains more 0s than 1s or if $i = 0$ then $xz$ contains fewer 0s than 1s.

We have derived a contradiction. Therefore $L$ is not regular. QED

3. Describe the error in the following “proof” that $0^*1^*$ is *not* a regular language. Note that there is an error because $0^*1^*$ is a regular language.

The proof is by contradiction. Assume that $L = 0^*1^*$ is regular and $p$ is the pumping length for $L$ given by the pumping lemma. Choose $w$ to be the string
You know that \( w \in L \) but \( w \) can not be pumped, since any \( xyyz \) will have more 0s than 1s. Thus you have a contradiction so \( 0^p1^p \) is not regular.

**Solution:** The error is in the statement \( w \) can not be pumped, since any \( xyyz \) will have more 0s than 1s.

The chosen string \( 0^p1^p \) has equal numbers of 0s and 1s and is in the language. If pumped then it loses the equal number of 0 and 1 property, but the resulting string \( \) is in the language \( L \) because it still has all 0s before 1s.

We need to find a string that can not be pumped in order to show \( L \) is not regular.

4. For the language \( L = \{a^ib^jc^k \mid i, j, k \geq 0 \land (i = 1 \rightarrow j = k)\} \)

(a) show that \( L \) is not regular
(b) show that \( w = a^ib^jc^k \) satisfies the pumping lemma conditions (for some \( i, j, k \))
(c) explain why parts a) and b) do not contradict the pumping lemma

**Solution:**
al) \( L \) is not regular because for \( w = a^ib^jc^k \) \( w \) can not be pumped. That is there is no possible \( xyz = w \) that can be pumped:
Suppose any \( |xy| \leq p \) and \( |y| \geq 1 \). Then either \( x \) is empty or \( ab^k \) and \( y \) is either \( ab^k \) or \( b^k \) for some \( k \leq p \). If \( y = ab^k \) then \( xyyz \notin L \) since it has as after b. If \( y = b^k \) then \( xyyz \notin L \) since it has more bs than cs.

b) \( L \) satisfies the pumping lemma conditions but only for some \( w \). For example, Let \( p \) be the pumping length and \( w = a^pb^jc^k \). Then for any \( xyz \) we have \( x \) and \( y \) contain only as and that \( xy^mz \in L \) for any \( m \geq 0 \) since the string still separates all its as, bs and cs.

c) No contradiction because pumping lemma is \( R \rightarrow P \) but if \( R = false \) then we can’t say anything about the truth of \( P \).

5. Describe a grammar that generates all binary strings that have equal numbers of 0s and 1s.

**Solution:**
This is version of the balanced brackets language. So a possible grammar is

\[
S \rightarrow \epsilon | 0S1 | 1S0 | SS
\]

6. Design a pushdown automata (PDA) and draw the state machine diagram for the language of all binary strings that have equal numbers of 0s and 1s.

**Solution:**
Strategy:
Put the $ symbol on the stack.
If input 1 when top of stack is 0 then we have a match, so pop the 0. If input 0 when top of stack is 1 then we have a match, so pop the 1.

Non-deterministically, if input 0 then push 0 onto the stack guessing that the top of the stack is also 0.
Non-deterministically, if input 1 then push 1 onto the stack guessing that the top of the stack is also 1.

If the stack is finished ($) and there is no more input then accept the string.

Pushdown automata:

7. Define a grammar that generates all binary strings with more 0s than 1s.

**Solution:**

Idea: start with a 0 and then build around it. The scaffolding around the initial 0 has either equal 0s and 1s or just 0, so there will always be at least one more 0 than 1s.

Grammar:

\[
S \rightarrow A0A \\
A \rightarrow AA \mid 1A0 \mid 0A1 \mid 0 \mid \epsilon
\]

8. Design a pushdown automata (PDA) and draw the state machine diagram for the language of all binary strings with more 0s than 1s.

**Solution:**

Strategy:
This is similar to the equal number of 0s and 1s machine in the last question. But now, guess when you reach the end of the inputs and move to a state for checking that there are only 0s left on the stack by popping them all off.

**FIX:**
the node before the accept node says "see 0s and do nothing with
them”.
but actually, for the transition to the accept state to work, we have
to get rid of those zeroes.
so fix that

ÂÁ: **FIX TO FIX**
Oh, actually that’s not necessary, because it’s non-deterministic.
The previous node lets us ”eat” as many 0s and 1s off the stack as
we like, we just have to guess when to transition and leave exactly
one on there, so the final transition will work.

\[
\begin{align*}
0, 1 & \rightarrow \epsilon; 1, 0 \rightarrow \epsilon \\
\epsilon, \epsilon & \rightarrow \$ \\
\epsilon, \epsilon & \rightarrow \epsilon \\
1, \epsilon & \rightarrow 1; 0, \epsilon \rightarrow 0 \\
\epsilon, \$ & \rightarrow \epsilon \\
\epsilon, 0 & \rightarrow 0 \\
\epsilon, 1 & \rightarrow \epsilon
\end{align*}
\]