Discrete Event Systems
Sample Solution to Exercise 4

1 Regular and Context-Free Languages

a) Sometimes, even simple grammars can produce tricky languages. We can interpret the 1s and 2s of the second production rule as opening and closing brackets. Hence, \( L(G) \) consists of all correct bracket terms where at least one 0 must be in each bracket.

Choose \( w = 1^p0^p \in L(G) \). Let \( w = xyz \) with \( |xy| \leq p \) and \( |y| > 0 \) (pumping lemma). Because of \( |xy| \leq p \), \( xy \) is in the first \( 1^p \) of \( w \). According to the pumping lemma, we should have \( xy^iz \in L \) for \( i \geq 0 \). However, by choosing \( i = 0 \) we delete at least one 1 and get a word \( w' = 1^k0^p \) where \( k < p \). \( w' \) is not in \( L \) since it has less 1s than 2s. \( w \) is not pumpable. Therefore \( L(G) \) is not regular.

b) Since every regular language is also context-free, we can choose an arbitrary regular language. For example, we can choose the language \( L = \{0^n1, n \geq 0\} \) which is clearly regular. The corresponding context-free grammar is \( S \rightarrow 0S | 01 \).

2 Context-Free Grammars

a) \[
S \rightarrow SAS \mid A \\
A \rightarrow 0 \mid 1
\]

Note: The language is regular!

b) One possible solution is to use three productions: A first one which guarantees that there is at least one ‘1’ more; a second one which produces all possible strings with the same number of ‘0’ and ‘1’; and finally, a production to add further 1’s at arbitrary places:

\[
S \rightarrow T1T \\
T \rightarrow T0T0T \mid T1T0T \mid U \\
U \rightarrow 1U \mid \epsilon
\]

3 Pushdown Automata

a) \( \varepsilon, 0, 00, () \)

b) It is unambiguous, i.e., there is a unique derivation tree for each word. Each word \( w \neq \varepsilon \) in \( L(G) \) contains a rightmost 0 or parenthesis expression ‘(S)’ that can be unanimously assigned to a \( A \) in each node of the derivation tree. Due to \( S \rightarrow SA \), each sequence of \( A \)'s is unambiguous.

c) A push-down automaton \( M \) is deterministic iff in each state, there is exactly one successor state for any combination \( (x, y) \in \Sigma \times \Gamma \) where \( \Sigma \) is the string input alphabet and \( \Gamma \) is the stack alphabet. Note that if a state \( q \) has only one outgoing transition ‘\( (\varepsilon, \epsilon) \rightarrow \$ \)’ the PDA is still deterministic since there is no ambiguity of what the successor state of \( q \) will be. If a state \( q \), however, has two outgoing transitions ‘\( (\varepsilon, \$) \rightarrow \$ \)’ and ‘\( (\varepsilon \rightarrow \$ \)’, it is unclear which transition the system should take if the string input in state \( q \) is ‘(’ and the top element on
the stack is ‘$’. As with deterministic FAs we take the liberty of omitting transitions leading to an (imaginary) fail state as well as the fail state itself when drawing deterministic PDAs. An instance of a deterministic PDA accepting $L(G)$ is the following:

If we would assume our PDA recognizes the end of the input string (denoted by ‘$’), the following deterministic pushdown automaton would also do the job:

Note that by replacing ‘$’ in the above PDA by ‘\sigma’ we get a correct non-deterministic PDA for $L(G)$.

4 Pumping Lemma revisited

a) Let us assume that $L$ is regular and show that this results in a contradiction.

We have seen that any regular language fulfills the pumping lemma. I.e. there is a $p$, such that for every word $u \in L$ with $|u| \geq p$ it holds that: $u$ can be written as $u = xyz$ with $|xy| \leq p$ and $1 \leq |y| \leq p$, such that $\forall i \geq 0 : xy^iz \in L$.

In order to obtain the contradiction, we need to show that there is at least one word $w \in L$ with $|w| \geq p$ for which it is not possible to form the string partition $w = xyz$, s.t. $|xy| \leq p$, $1 \leq |y| \leq p$, and $\forall i \geq 0 : xy^iz \in L$.

First, we need to overcome the problem that we do not know the value of $p$. The standard trick is to consider words whose length depends on $p$. E.g. consider the word $w = 1p^2 \in L$.

By the pumping lemma, we can write $w = 1p^2$ as $xyz$. What remains to show is that there is no partition $xyz$ that satisfies $|xy| \leq p$, $1 \leq |y| \leq p$, and $\forall i \geq 0 : xy^iz \in L$.

The expression $w = xy^iz$ can be written as $xy^iz = 1|x|1|y|1|z|$. Because $|w| = p^2$, we know that $|z| = p^2 - |x| - |y|$, and therefore, $xy^iz = 1|x|1|y|1p^2 - |x| - |y| = 1p^2 + (i-1)|y|$.

To obtain the contradiction, we need to find an $i \geq 0$, such that $xy^iz \notin L$. For example, consider $i = 0$. Then we have $w^0 = xy^0z = 1p^2 - |y|$. Clearly, $|w^0| < p^2$, as $|y| \geq 1$. Note that we argue independent of the partition $w = xyz$, we do not pick a specific $x$ and $y$ and therefore the following holds for all possible partitions.

If $w^0 \in L$, then $|w^0|$ is a square number, smaller than $p^2$. But the next smaller square number, $(p - 1)^2$, is strictly smaller than $|w^0|$: $(p - 1)^2 = p^2 - 2p + 1 < p^2 - |y| = |w^0|$, which shows that $|w^0|$ cannot be a square number. This shows that there is no partition for $w$ that allows to fulfill the pumping lemma conditions. But this should be the case if $L$ is regular. Thus, we have a contradiction, which concludes the proof.
b) Consider the alphabet $\Sigma = \{a_1, a_2, ..., a_n\}$ and the language $L = \bigcup_{i=1}^{n} a_i^*$. The language is regular, as it is the union of regular languages, and the smallest possible pumping number $p$ for $L$ is 1. But any DFA needs at least $n + 1$ states to distinguish the $n$ different characters of the alphabet. Thus, for the DFA, we cannot deduce any information from $p$ about the minimum number of states.

The same argument holds for the NFA.