Discrete Event Systems
Solution to Exercise 4

1 Regular and Context-Free Languages

- 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.

  $L(G)$ is not regular. Choose $x = 1^n 0 2^n \in L(G)$. Let $x = uvw$ with $|uv| \leq n$ and $|v| > 0$ (pumping lemma). Because of $|uv| \leq n$, uv is in the first 1^n of x. According to the pumping lemma, we have $uv^i w \in L (i \geq 0)$. If we choose $i = 0$ we get $1^k 0 2^n \notin L (k < n)$.

- Since *every* regular language is also context-free, we can choose an arbitrary regular language. For example, we can choose the language $L = \{0^n 1, n \geq 1\}$ which is clearly regular. The corresponding context-free grammar is $S \rightarrow 0S \mid 1$.

2 Context-Free Grammars

- $$S \rightarrow SAS \mid A \quad , \quad A \rightarrow 0 \mid 1.$$

- 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:

$$\begin{array}{rl} S \rightarrow & T1T \\ T \rightarrow & T0T1T \mid T1T0T \mid U \\ U \rightarrow & 1U \mid \epsilon \end{array}$$

3 Pushdown Automata

a) ϵ, 0, 00, ()

b) It is unambiguous, i.e., there is a unique derivation tree for each word. Each word $w \neq \epsilon$ 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 As is unambiguous.

c) The following deterministic pushdown automaton does the job:

4 Counter Automaton

- A counter automaton is basically a finite automaton augmented by a counter. For every regular language $L \in L_{reg}$, there is a finite automaton A which recognized L. We can construct a counter automaton C for recognizing L by simply taking over the states and transitions of A and *not* using the counter at all. Clearly C accepts L. This holds for every regular language and therefore, $L_{count} \in L_{reg}$.

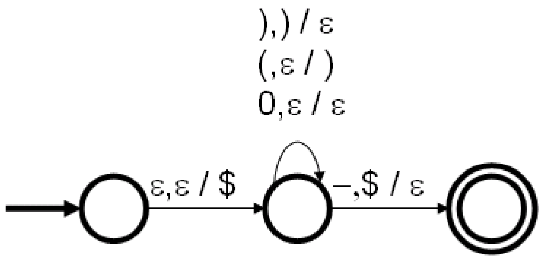

Figure 1: Pushdown automaton accepting $L(G)$

- Consider the language L of all strings over the alphabet $\Sigma = \{0, 1\}$ with an equal number of 0s and 1s. We can construct a counter automaton with a single state q that increments/decrements its counter whenever the input is a 0/1. If the value of the counter is equal to 0, it accepts the string. Hence, L is in L_{count}.

  On the other hand, it can be proven (using the pumping lemma) that L is not in L_{reg} and it therefore follows $L_{count} \notin L_{reg}$.

- First, we show that a pushdown automaton can simulate a counter automaton. Hence, PDA's are at least as powerful as CA's! The simulation of a given CA works as follows. We construct a PDA which has exactly the same states as the CA. The transitions also remain between the same pairs of states, but instead of operating on a INC/DEC register, we have to use a stack. Concretely, we store the state of the counter on the stack by pushing '+' and '-' on the stack. For instance, a counter value '3' is represented by three '+' on the stack, and similarly a value '-5' by five '-'. Therefore, when the CA checks whether the counter equals 0, the PDA can check whether its stack is empty.

  In the following, we give just one example of how the transitions have to be transformed. Assume a transition of the counter automaton which, on reading a symbol s increments the counter—independently of the counter value. For the PDA, we can simulate this behavior with three transitions: On reading s and if the top element of the stack is '-', a minus is popped; if the top element is a '+', another '+' is pushed; and if the stack is empty, also a '+' is pushed.

  Hence, we have shown that the PDA is at least as powerful as the CA, and it remains to investigate whether both CA and PDA are equivalent, or whether a PDA is stronger. Although it is known that the PDA is actually more powerful, the proof is difficult: There is no pumping lemma for CA's for example such that we can prove that a given context-free language cannot be accepted by a CA. However, of course, if you have tackled this issue, we are eager to know your solution... :-)