ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Networked Systems Group (NSG)

HS 2019

Prof. L. Vanbever & T. Holterbach, & M. Apostolaki.
based on Prof. R. Wattenhofer's material

Discrete Event Systems

Solution to Exercise Sheet 5

1 Counter Automaton

a) A counter automaton is basically a finite automaton augmented by a counter. For every regular language $L \in \mathcal{L}_{reg}$, there is a finite automaton A which recognizes 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, $\mathcal{L}_{reg} \subseteq \mathcal{L}_{count}$.

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

Some languages where the (non-finite) frequency of one or several symbols depends on the frequency of other symbols can be recognized by counter automata. Such languages cannot be recognized by finite automata.

c) First, we show that a pushdown automaton can simulate a counter automaton. Hence, PDAs are at least as powerful as CAs! 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 an 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 CAs 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... :-)

2 Designing Turing Machines

The proposed Turing machine decrements the value of a until $a = 0$. In each step, it adds a '1' to the output:

1. Move the TM head to the right of a and place a \$ sign. We will use this marker to return to the LSB of a.
2. Look at the LSB of a. If it is '1', we change it to 0 (transition between q_1 and q_3) and move to the right. Then, we continue moving to the right until we hit a $\square$, which is changed to a '1' (transition q_4 to q_5). Finally, we move back to the LSB of a.
3. If the LSB of a is '0', we search for the first '1' in a from the right (loop on q_1 and transition from q_1 to q_3).

3.1 If we find a '1', we change it to '0'. While moving back to the \$ symbol, we change all '0' to '1' (self-loop on q_3). Then, we proceed as in point 2 after passing the \$ symbol.

3.2 If we don't find a '1' in a at all (transition q_1 to q_6), we start the cleanup procedure: Remove all 0 on the right of the \$ symbol, and finally remove the \$ symbol itself and move to the right of u.