University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour

Theoretical Computer Science - Bridging Course
Sample Solution Exercise Sheet 5

Due: Tuesday, 27th of May 2025, 12:00 pm

Exercise 1: Proving NonCFL *(5+5 Points)*

Use the Pumping Lemma to show that the following languages are not CFL.

1. $L = \{a^n b a^{2n} b a^{3n} \mid n \geq 0\}$
2. $L = \{a^i b^j c^k \mid i < j \text{ and } i < k\}$

Bonus: $L = \{a^m \mid m \text{ is a prime}\}$
NB: If you wish you can try first and prove it nonregular using the Pumping Lemma for regular languages and the same idea should be extended to CFLs.

Sample Solution

In all parts, we are going to **assume** that L is a CFL. This means that the property from the Pumping Lemma should hold **true** for L (i.e. there exists a pumping length $p \geq 1$ such that for every string $w \in L$ where $|w| \geq p$, we can break w into five substrings $w = uvxyz$ such that properties 1, 2, and 3 hold true). **But**, we will show that in fact this property will **not** hold **true** for L (i.e. we will show that: for this pumping length $p \geq 1$, there exists a (bad) word $w \in L$ of length at least p such that for every composition of $w = uvxyz$ that satisfies properties 2 and 3, it turns out property 1 is never satisfied). This will then give us a **contradiction** to the Pumping Lemma, hence, our initial assumption must have been wrong. Therefore, L is a nonCFL.

1.
   - Assume L is a CFL.
   - This means that the property from the Pumping Lemma for context free languages should hold true for L (you can state the pumping lemma here).
   - Now, let p be any pumping length.
   - Consider a string $w := a^p b a^{2p} b a^{3p} \in L$ and has length at least p.
   - Consider any composition $w = uvxyz$ with $|vy| > 0$ and $|vxy| \leq p$ (conditions 2 and 3 of the Pumping Lemma).
   - We show that uv^2xy^2z cannot be in the language, giving a contradiction.
     Indeed, we have two cases:
     Case 1: If v or y contained b, then the string uv^2xy^2z would have more than two b's and is therefore not in L.
     Case 2: If neither v nor y contains b, then that means that v as well as y is fully contained in one of the three segments a^p, a^{2p} and a^{3p}. But then pumping w up to uv^2xy^2z would violate the $1:2:3$ length ratio of the segments, because the length of at least one segment is changed (as $|vy| > 0$) and at least one segment keeps its length. Thus, uv^2xy^2z is not in L.
   - This is a contradiction to the Pumping Lemma.

- Therefore, L is a nonCFL.

2.
   - Assume L is a CFL.
   - This means that the property from the Pumping Lemma for context free languages should hold true for L (you can state the pumping lemma here).
   - Now, let p be any pumping length.
   - Consider a string $w := a^p b^{p+1} c^{p+1} \in L$ and has length at least p.
   - Consider any composition $w = uvxyz$ with $|vy| > 0$ and $|vxy| \leq p$ (conditions 2 and 3 of the Pumping Lemma).
   - We show that uv^2xy^2z or uv^0xy^0z cannot be in the language, giving a contradiction. Indeed, we can be in either of the four cases:
     Case 1: If v or y contains a mixture of symbols from $\{a, b, c\}$; then uv^2xy^2z would be in the wrong order and thus not in L.
     Case 2: If vxy in contained in a^p, thus v or y consists entirely of 'a's [1]. Then if we pump up e.g. uv^2xy^2z, there is no way for v and y, which cannot have both 'b's and 'c's to generate enough of these letters to keep their number greater than that of the 'a's (it can do it for one or the other of them, but not both). Thus, uv^2xy^2z is not in L.
     Case 3: If v or y contains only 'b's or only 'c's. Consider the string uv^0xy^0z which must be in L. Since we have dropped both v and y, we must have at least one 'b' or one 'c' less than we had in $uvxyz$, which was $a^p b^{p+1} c^{p+1}$. Consequently, this string no longer has enough of either 'b's or 'c's to be a member of L. Thus, uv^0xy^0z is not in L.
     Case 4: If v or y contains only 'c's. Similar argument as in case 3.
   - This is a contradiction to the Pumping Lemma.
   - Therefore, L is a nonCFL.

Bonus solution: Assume L is a CFL. This means that the property from the Pumping Lemma for context free languages should hold true for L. Now, let p be any pumping length. Let $t > p$ be a prime. Consider the string $w = a^t \in L$ and of length at least p. Let $uvxyz$ be any decomposition of w^t, satisfying that $|vy| > 0$ and $|vxy| \leq p$. Thus v must be a^i and y must be a^j, for some $i \geq 0$ and $j \geq 0$ such that $i + j > 0$. If we pump up $t + 1$ times, then we have the string $uv^{t+1}xy^{t+1}z = uvxyz \cdot v^t y^t = a^{t(1+i+j)}$. Since both t and $1 + i + j$ are greater than 1, then $t(1 + i + j)$ is not a prime. Hence, $uv^{t+1}xy^{t+1}z$ is not in L. Thus a contradiction to the Pumping Lemma. Therefore, L is nonCFL.

Remark: L can also be proven non regular using the pumping lemma for regular languages with a simialr argument as above. Indeed, assume the language is regular. This means that the property from the Pumping Lemma for regular languages should hold true for L. Let p be the pumping length. Let $t > p$ be a prime. Consider $w = a^t$. Then, any composition of a^t, satisfying that $|y| > 0$ and $|xy| \leq p$ should like the following: let $x = a^i$, $y = a^j$ and $z = a^k$ such that $i+j+k = t$, $j > 0$, and $i+j \leq p$. We pump up as above and get $xy^{t+1}z = a^i a^{j(t+1)} a^k = a^{(i+j+k)} a^{tj} = a^{t(j+1)}$, where both t and $j + 1$ are greater than 1. Therefore, $t(j + 1)$ is not a prime, and hence $xy^{t+1}z$ is not in L. Thus a contradiction to the Pumping Lemma. Therefore, L is not regular.

Exercise 2: Operation Shift *(3+3 Points)*

Consider a Turing machine $\mathcal{M}$ that is given an arbitrary input string over alphabet $\Sigma = \{1, 2, \ldots, n\}$ on its input tape. We would like $\mathcal{M}$ to insert an empty cell, i.e., $\sqcup$, at the beginning of the tape without

[1] We say (v *or* y) and not (v *and* y), since condition 2 is still satisfied even if either v or y was an ε. So a possible composition of w in here (case 1) can have one of them as ε, so e.g. $v = \varepsilon$ and $y = a$. Thus, we need to make sure that we have covered/talked about these scenarios too

removing any symbol on the tape. As an example, the Turing machine is supposed to change the input tape of the form $\langle 2, 4, 4, 6, 1, 8, 4, \sqcup, \sqcup, \dots \rangle$ to $\langle \sqcup, 2, 4, 4, 6, 1, 8, 4, \sqcup, \sqcup, \dots \rangle$. Although this operation is not explicitly defined for a Turing machine, one can consider such an operation as shifting the whole string one cell to the right on the input tape.

(a) Give a formal definition of $\mathcal{M}$ to perform the desired operation such that $\mathcal{M}$ recognizes the language Σ^*.

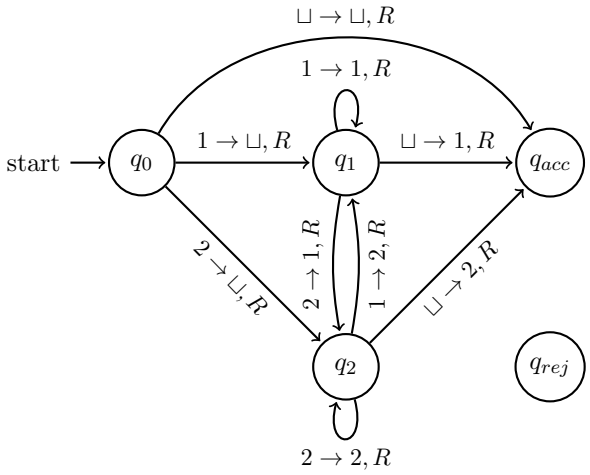
(b) For $n = 2$, i.e., $\Sigma = \{1, 2\}$, draw the state diagram of your constructed Turing machine.

Sample Solution

(a) Consider the set of states to be $Q = \{q_0, q_{acc}, q_{rej}\} \cup \{q_c | c \in \Sigma\}$. And Γ is the tape alphabet where $\sqcup \in \Gamma, \Sigma \subseteq \Gamma$. Then, the transition functions are as follows.

$$\begin{aligned}
&\forall c \in \Sigma;\ \delta(q_0, c) = (q_c, \sqcup, R) \\
&\forall c, c' \in \Sigma;\ \delta(q_c, c') = (q_{c'}, c, R) \\
&\forall c \in \Sigma;\ \delta(q_c, \sqcup) = (q_{acc}, c, R) \\
&\delta(q_0, \sqcup) = (q_{acc}, \sqcup, R)
\end{aligned}$$

(b)



Remark: *As discussed in class, if we want to be more precise/correct and for the state diagram to be well defined, we can redefine the trasnsition function of the TM from the lecture to be e.g* $\delta : Q \backslash \{q_{acc}, q_{rej}\} \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

Remark: Whenever needed in the upcoming exerices, we can detect the beginning of the input on the tape e.g. by initially shifting all the input one step to the right (cf. exercise 1). Hence, we can always have a blank symbol placed at the leftmost of the tape.
Alternatively, one can use a trickier method mentioned from the book by Sipser: *"A trickier method of finding the left-hand end of the tape takes advantage of the way that we defined the Turing machine model. Recall that if the machine tries to move its head beyond the left-hand end of the tape, it stays in the same place. We can use this feature to make a left-hand end detector. To detect whether the head is sitting on the left-hand end, the machine can write a special symbol over the current position while recording the symbol that it replaced in the control. Then it can attempt to move the head to the left. If it is still over the special symbol, the leftward move didn't succeed, and thus **the head must have been at the left-hand end**. If instead it is over a different symbol, some symbols remained to the left of that position on the tape. Before going farther, the machine must be sure to restore the changed*

symbol to the original."

Remark: For the rest of course and whenever needed, we detect the beginning of the input on the tape by e.g. initially shifting all the input one step to the right. Hence, a blank symbol will be placed at the leftmost of the tape.

Exercise 3: Constructing a TM *(4 Points)*

Consider alphabet $A = \{1, 2, \ldots, 9\}$. We call a string S over A a *blue* string, if and only if the string consisting of the odd-positioned symbols in S is the reverse of the string consisting of the even-positioned symbols in S. For example $S = 14233241$ is a blue string since the substring of the odd-positioned symbols is 1234 which is the reverse of the substring of the even-positioned symbols, i.e., 4321.

Design a Turing machine which accepts all blue strings over A. You do not need to provide a formal description of the Turing machine but your description has to be detailed enough to explain every possible step of a computation.

Sample Solution

A high level description would be as follows:
On input S, first go through all symbols. If it is an odd number, reject. Else repeat the following:
Go left until you reach the first unmarked symbol, mark it, go right to the last unmarked symbol, mark it, and compare both symbols. If they are different, reject.
Accept if all symbols are marked.

Exercise 4: Bonus Question *(Points)*

Consider a push-down automata with two stacks instead of one. Can it be as powerful as a Turing machine? Give an intuitive explanation?
Hint: Try to show how can the Two-Stack PDA simulate the TM tape.

Sample Solution

We first add the entire input to the first stack, then reverse it so it is in the second stack with the first character at the top. Now, moving the tape head left or right is the same as popping a symbol off of one stack and pushing it onto the other (and in this way you can see that, the first stack holds what is to the left of the head, while the second stack holds what is to the right of the head along with the symbol which is under the current head).
Moreover, reading the symbol under the tape head is the same as reading the symbol at the top of the second stack. And writing a new symbol under the tape head is emulated by replacing the symbol on top of the second stack with the new symbol (i.e. popping the symbol from top of the stack then pushing this new symbol onto it). This allows to simulate the tape of TM , where the TM head is corresponding to the heads of the two stacks of the Two-Stack PDA .

Note: We can use this to show that a push-down automata with two stacks is equally powerful as Turing machines (i.e. with regards to language recognition).