18.404/6.840 Recitation 3

Abbas Zeitoun

September 18, 2020

Exercise 1

Let $A = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k | i > j > k\}$. Show that A is non-context-free.

Solution: Assume that A is context-free. Then according to the pumping lemma for CFLs, A has a pumping length p such that $\forall s \in A$, if $|s| \geq p$, then s may be divided into 5 pieces $s = uvxyz$ satisfying the conditions:

1. for each $i \geq 0$, $uv^ixy^iz \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$

Consider the string $s = \mathtt{a}^{p+2}\mathtt{b}^{p+1}\mathtt{c}^p \in A$, and consider the different values that v and y can take when we divide s into 5 parts $s = uvxyz$. Note that the third condition of the pumping lemma states that $|vxy| \leq p$, which means that vy cannot contain all three symbols `a`, `b` and `c` at the same time. This restricts the values that v and y can take to the following cases:

Case 1: vy consists of only "`a`" symbols:
Consider the string $s' = uv^ixy^iz$. If we set $i = 0$ (i.e. if we pump down), s' will contain at most $p + 1$ "`a`" symbols while still containing $p + 1$ "`b`" symbols. So $s' \notin A$, which violates the first condition of the pumping lemma.

Case 2: vy consists of only "`b`" symbols:
Consider the string $s' = uv^ixy^iz$. If we set $i = 0$ (i.e. if we pump down), s' will contain at most p "`b`" symbols while still containing p "`c`" symbols. So $s' \notin A$, which violates the first condition of the pumping lemma.

Case 3: vy consists of only "`c`" symbols:
Consider the string $s' = uv^ixy^iz$. If we set $i = 2$ (i.e. if we pump up), s' will contain at least $p + 1$ "`c`" symbols while still containing $p + 1$ "`b`" symbols. So $s' \notin A$, which violates the first condition of the pumping lemma.

Case 4: vy consists of both "`a`" and "`b`" symbols:
There are several ways to divide the "`a`"s and "`b`"s between v and y, but since the argument we'll use only depends on the counts of the symbols in s', we can group all these divisions into a single case.
Consider the string $s' = uv^ixy^iz$. If we set $i = 0$ (i.e. if we pump down), s' will contain at most p "`b`" symbols while still containing p "`c`" symbols. So $s' \notin A$, which violates the first condition of the pumping lemma.

Case 5: vy consists of both "`b`" and "`c`" symbols:
Consider the string $s' = uv^ixy^iz$. If we set $i = 2$ (i.e. if we pump up), s' will contain at least $p + 2$ "`b`" symbols while still containing $p + 2$ "`a`" symbols. So $s' \notin A$, which violates the first condition of the pumping lemma.

So for all the different values that v and y can take, s can be pumped in a way that violates the pumping lemma. So A is not a context-free language.

Exercise 2

A 2-PDA is a pushdown automaton that has 2 stacks. Show that 2-PDAs can recognize the same class of languages as TMs.

Solution: Given a TM T that recognizes a language A, we would like to construct a 2-PDA P that recognizes A. P will make use of its two stacks to simulate T and return the same result on any input string s. The simulation can occur as follows:

1. P reads the entire input string and pushes the read symbols one by one onto its first stack S_1.
2. P pops the symbols from S_1 one by one and pushes them onto the second stack S_2. Once S_1 is empty, P pops the topmost element of S_2 and pushes it onto S_1. At this point, S_1 will contain only the first symbol of the input string, and S_2 will contain all the remaining symbols starting from the second symbol (at the top of S_2) to the last symbol (at the bottom of S_2).
3. With the entire input string now on S_1 and S_2, P can use its finite control to simulate T, using its two stacks to simulate the tape of the TM. The position of the head of the simulated TM will be the topmost element of S_1.
4. To read the symbol under the current position of the head, P can read the topmost element of S_1.
5. To write a symbol under the current position of the head, P can pop the topmost element of S_1 and push in its place the new symbol.
6. To move right along the tape, P can pop the topmost element of S_2 and push that onto S_1. If S_2 is empty, P pushes the blank "␣" symbol onto S_1 instead.
7. To move left along the tape, P can pop the topmost element of S_1 and push that onto S_2. If S_1 becomes empty after this operation, we reverse it by popping the topmost element of S_2 and pushing that back onto S_1. This makes sure that the head never goes past the beginning of the TM's tape.
8. The last point we need to take care of in the simulation is the edge case when the input string is empty. After reading its input string, P can check if S_1 is empty, and if it is, P can push onto S_1 the blank "␣" symbol.

Given that P can simulate the behavior of T on any input string s, P can return the same result as T for any such s. So if T recognizes A, P will recognize A as well.

Exercise 3

A reset-TM is a Turing machine that does not have the ability to move its head to the left along the input tape. Instead, a reset-TM has the ability to reset the position of its head to the beginning of the input tape. Show that reset-TMs and TMs recognize the same class of languages.

Solution: We would like to show that a reset-TM R recognizes a language A iff a TM T recognizes A. We can show this by letting one kind of machine simulate the other kind of machine and return the same result as the simulated machine.

($\longrightarrow$): If a reset-TM R recognizes a language A then a TM T recognizes A.
This direction is straightforward to prove. T can simulate R on its input string and return the same result as R on that string. To simulate R, T can perform the exact same operations as R, and when R uses the "reset" operation, T can simulate that by moving left repeatedly until it reaches the beginning of the string. Since T can use this simulation to replicate the behavior of R on any input string, if R recognizes A, T will recognize A as well.

($\longleftarrow$): If a TM T recognizes a language A then a reset-TM R recognizes A.
This proof is similar to the proof of the other direction. R can simulate T on its input string and return the same result as T on that string. R can directly simulate all of the basic operations that T can perform, except for moving the head one step to the left, so if we can show how to simulate that, the proof will be done.

One approach we could try is to expand the alphabet of R to include a marked/dotted symbol for each symbol in the alphabet. R can then use the new alphabet to mark the cell just to the left of the current position of the head. After that, R can reset the position of the head and then move right until it finds the marked cell. That cell would be the cell just to the left of the original position of the head.

The one problem with this approach is that R cannot mark the cell one step to the left without moving its head to that cell, and R cannot move its head one step to the left to accomplish that. To remedy this, R can achieve the same behavior by marking the current cell, and then shifting all of the tape's contents one step to the right *without moving the position of the mark*. If R now resets the position of the head to the beginning of the tape and moves right until it finds the marked cell, that cell would be the cell just to the left of the original position of the head.

Finally, we note that R can shift the tape contents to the right by one step without needing to move its head to the left. It can do so as follows:

- For each symbol a in the alphabet, create a state q_a. (This is done when R is being constructed, not while it's running).
- Move to the beginning of the tape and read the value of the first cell. Let this value be b. Make a transition to state q_b and move the head one step to the right. All of this can be done in one step of computation, according to the formal definition of a Turing machine.
- Now R can repeat the following until it reaches the end of the string:
  - Read the value c of the current cell.
  - Let q_i be the state that the TM is currently in. Write i in the current cell and make a transition to state q_c. Again, the head moves one step to the right, and again, all of this is done in one step of computation.

Of course, some accounting still needs to be done to take care of minor implementation details (keeping track of the beginning and the end of the string on the input tape, accounting for marks when using the above procedure to shift over the tape contents, etc), but those are relatively straightforward to deal with.

To conclude, R can use the simulation procedure described above to replicate the behavior of T on any input string, so if T recognizes A, R will recognize A as well.