CS 208 : Automata Theory and Logic 14-03-2024

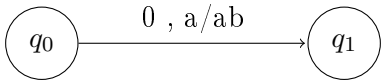
Lecture - 25

Topic: PushDown Automaton

Scribed by: Kavin Arvind (22b1019)
Checked and compiled by:

Representation

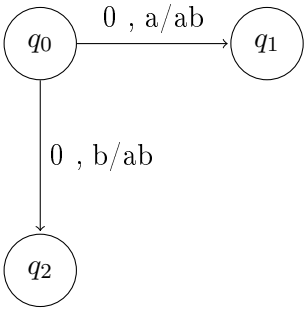
Let us try to equip stack along with the normal automaton which we have been seeing till now. Imagine a we an automaton, and there also exists a stack. We push and pop in the stack as we move in the automaton through an edge. So let's see its representation before formally defining.



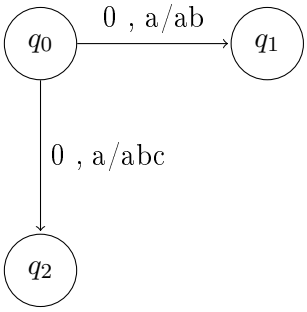
So here, it means that when we are at state q_0, and get an input alphabet as 0, and if the top of the stack is a, we pop out a from the stack and insert ab (i.e, we push b first, and push a next) before we go to the state q_1.

Defenition

We define an automaton $A = (Q, \sum, \tau, q_0, z_0, \delta, F)$. The defenition is very similar to what we have seen before. Just τ and z_0 are new to us. So here, τ represents the set of letters contained in the stack. In the above example, a and b were contained by τ whereas the alphabets were 0 and 1. So, here we can note that we don't store the alphabets that are inputted. It may belong to a totally new set τ. During the start, i.e at the start state our stack won't be empty and it would contain exactly 1 element and that is z_0 which belongs to τ.
Lets see how do we include non-determinism over here.



Deterministic as only 1 path is possible based on the top element of the stack



Non-deterministic as the top element of the stack being a, there are two paths

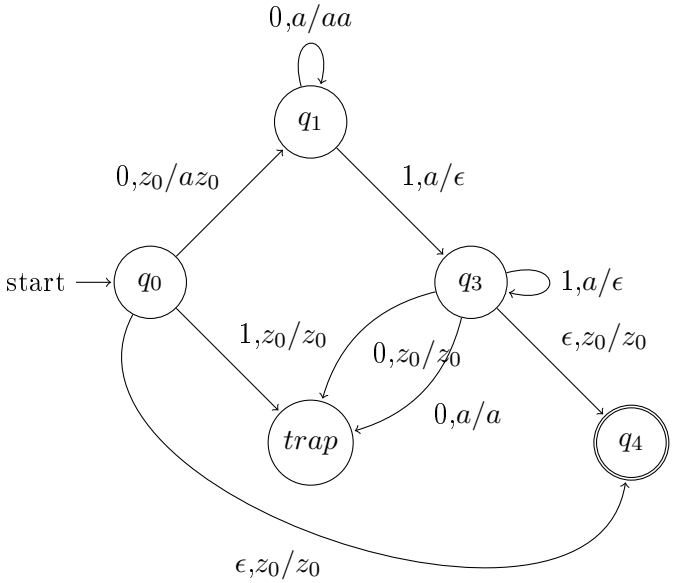
So here, the funtion δ can be defined as:

$$\delta : Q \times (\sum \cup \{\epsilon\}) \times \tau \to 2^{Q \times \tau *}$$

Remember that for our previous normal automaton, the defenition was $\delta : Q \times (\sum \cup \{\epsilon\}) \to 2^Q$. The difference is that, from every state, we have the input string, and top element of the stack we want to read. Thus its $Q \times (\sum \cup \{\epsilon\}) \times \tau$. From there, we get choise of pushing any string from $\tau *$ and so its $2^{Q \times \tau *}$.

An Example

Let's see how can we build pushdown automaton for the language $\{0^n 1^n | n \geq 1\}$.
Lets take $\tau = \{z_0, a\}$ and $\sum = \{0, 1\}$. Observe the below automaton for this language.



Here, we start at q_0, and go to q_1 when it starts with 0 and also we push a into stack. Basically, here the number of a's in the stack represent the number of 0's we've encountered so far. You can see that the self edge in q_1 pops a, and pushes two a's effectively incrementing the number of a's in the stack by 1. After that, as we start encountering 1's, we move to q_3 state and also we pop out a's untill all a's are done and we are left with z_0. If something wrong happens, we go to the *trap* state, and finally when we are done popping all a's, we would be left with just z_0, and we go to an accepting state q_4. After coming to the accepting state, if any letter is encountered, we would be struck and so we don't accept that word anyway.

So, by equiping automaton with a stack allows us to model many more languages which were earlier not able to be modeled. This type of automaton is called as **PUSHDOWN AUTOMATON (PDA)**.

Now, lets try to construct the automaton for the language $\{0^n 1^m | n \geq m, m \geq 1\}$. In the previous example itself, just make q_3 as accepting, and we're done.

Ways of Acceptance

Now, we see that there are two ways to accept a word by PDA. They are:-

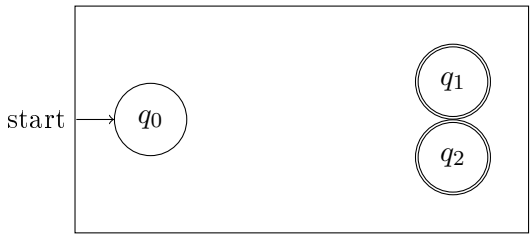
- Acceptance by final state: This is same as the traditional way we were following till now.
- Acceptance by empty stack: Whichever word makes the stack empty will be accepted and others won't. Language accepted this way is represented by $N(A)$ where A is the PDA.

Now, we want to show that both the methods are equivalent. To show this we have to show the following.
Given a PDA A_1,

1. Does there exists a PDA A_2 such that, $L(A_1) = N(A_2)$.
2. Does there exists a PDA A_3 such that, $N(A_1) = L(A_3)$.

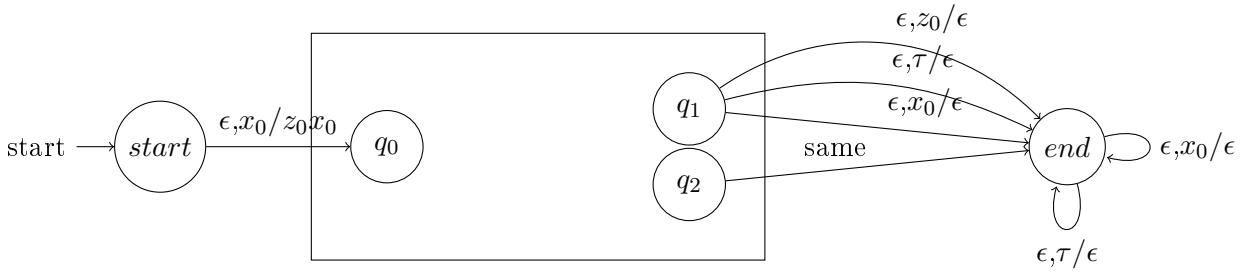
Does there exists a PDA A_2 such that, $L(A_1) = N(A_2)$?

Yes. We can construct such a PDA A_2 using A_1.



Let this be PDA $A_1 = (Q, \sum, \tau, q_0, z_0, \delta, F)$

Consider the below PDA A_2 with its stack variables $\tau \cup \{x_0\}$ where $x_0 \notin \tau$ and initial value in the stack as x_0.



Constructed PDA A_2

So here, we took x_0 which is not in τ and we have it initially in out stack. As we come to q_0, our stack contains x_0 at the bottom, and z_0 above. So now, the stack and everything resembles A_1, just that x_0 exists at the bottom of the stack and its just the same. We observe that, none of the edges inside the rectagle can remove x_0 from the stack as they work considering just τ being in stack. So, for words being accepted in A_1, they reach q_1 and q_2, and then, we pop out all elements in the stack as we go to *end* state, and end up accepting them in $N(A_2)$. We can also tell that if the stack is being emptied, then it reaches the state *end*(other states can't empty the stack) which should

have passed though q_1 and q_2 and so it would have been accepted by $L(A_2)$. Thus, we can tell that $L(A_1) = N(A_2)$. Hence Proved.