Language and Automata, Assignment 2

Krzysztof Rudnicki
Student number: 307585

January 9, 2022

Chapter 1

Regular expression

We are given following regular expression:

$$0^n 1^m 2^m 3^n$$

Chapter 2

Diagram of the automata

Draw a diagram of the automata (specifying states and transitions in the diagram).

I used the following notation for transitions: I, S / N_S Where:
I - Input from the string
S - Element currently on top of the stack
/ - a dash to make it easier to differentiate between *before* transition and *after*
N_S - How the top of the stack will look after the transiton, if there are 2 symbols read it like this: leftmost symbol will be on top of the stack as a new symbol, rightmost symbol is just below this symbol, I use this to denote that I am NOT deleting the earlier symbol from stack, if there is 1 symbol it means that this symbol will be on top of the stack, if there is ϵ it means that I pop (remove) the symbol from top of the stack.

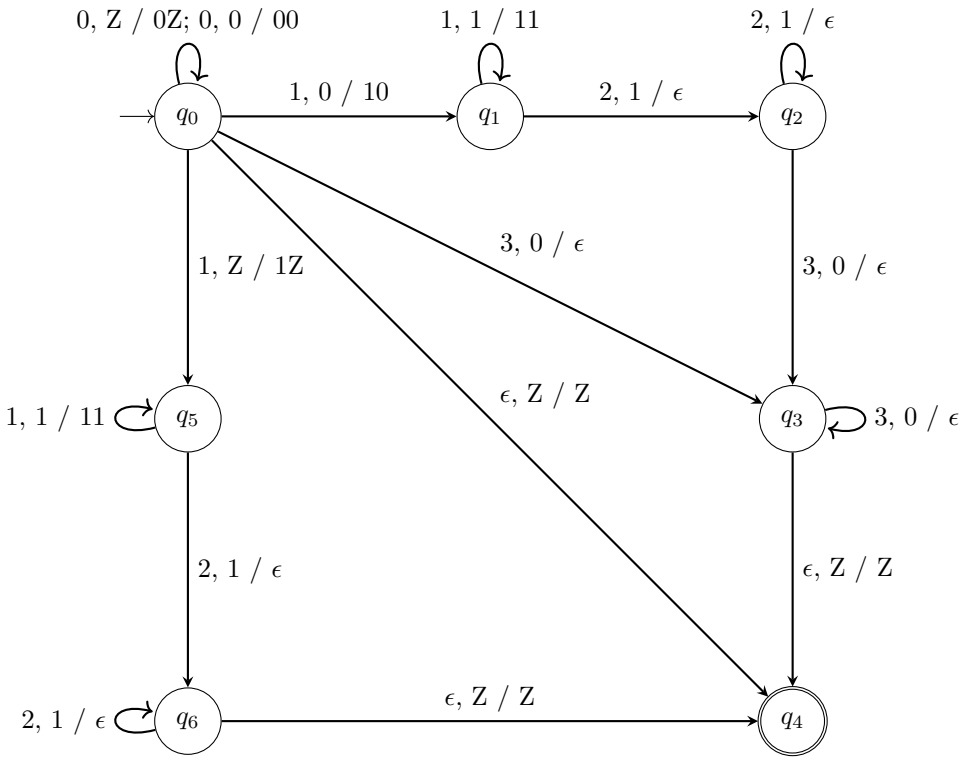

Figure 2.1: Diagram of the automata

Chapter 3

Description of states and the behaviour of PDA

Describe states and the behavior of PDA (push-down automata) in informal way [describe in words what's the meaning of each state and how the PDA behaves].

3.1 Cases

I recognized 4 possible cases for this automata:

1. m = 0 Our input string will look like this: 0^n3^n. If we encounter something like this we should keep pushing 0's to the stack until we encounter 3. Once we receive 3, we check if on top of the stack we have 0 and then pop it from stack. Then we can either empty the stack and run out of 3's and which point we go to the accepting state, otherwise we go to the fail state, we go to the fail state if:
   We run out of 3's before we run out of 0 (number of 3 is smaller than number of 0)
   We run out of 0's before we run out of 3 (number of 3 is bigger than number of 0) I will refer to the

   $$q_0 \rightarrow q_3 \rightarrow q_4$$

   path, as the case 1 route

2. n = 0 We will do exactly the same as in example above but swap 0 with 1 and 3 with 2. I will refer to the

   $$q_0 \rightarrow q_5 \rightarrow q_6 \rightarrow q_4$$

   path, as the case 2 route

3. n, m > 0 Our input string will look like this: $0^n1^m2^m3^n$. First we push 0's and 1's on stack. Once we encounter 2 we check if we have 1 on top of the stack, then pop it. We keep popping 1's until we run out of 2's. Then once we encounter 3 we check if we have 0's on top of the stack and keep popping 0's until we run out of 3's. If stack is empty by the end of the string we move to the accepting state, otherwise we move to the fail state.

   I will refer to the

   $$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4$$

   path, as the case 3 route

4. n = 0, m = 0 String will be empty but also accepted by this grammar. In this case we will just go straight to the accepting state. I will refer to the

   $$q_0 \rightarrow q_4$$

   path, as the case 3 route

3.2 Notes on behaviour and design decisions

I decided to make non-deterministic PDA since it was much easier as it allowed me to deconstruct the problem on the case basis, if you take a look at the "route" below the starting state, this is the route that corresponds to case number 2, where n = 0. This was very easy to implement since I used non-deterministic PDA.
I also decided to make this PDA accepting by non-empty stack, I use the letter 'Z' to denote the accepting symbol on stack, notice that when:
I get 0 at the start state I put this 0 AND 'Z' on stack - case 1 and 3 Later when I get epsilon from input I check if I am on 'Z' on stack and go to the accepting state
I get 1 at the start state I put this 1 AND 'Z' on stack - case 2 Later when I get epsilon from input I check if I am on 'Z' on stack and go to the accepting state
I get *epsilon* at the start state I put 'Z' on stack (and go to the final state) - case 4
I decided to use PDA accepting by non-empty stack since it is more similar to how actual stacks in computers work, like stack used by x86 assembly language I am familar with.

3.3 Description of states

- q_0 - Starting state, based on the input we get from string it decides whether we go down the case 1 route (0 then 3 on input), case 2 route (1 on input), 3 route (0 on input) or case 4 route (ϵ on input).
- q_1 - State in which we keep on pushing 1 on stack from input, once we receive 2 we move onto state q_2

- q_2 - State in which we keep pushing 2 on stack from input, once we receive 3 we move onto state q_3
- q_3 - State in which we keep *popping* 0 from stack if we receive the 3 from input, once the input is empty **AND** we have accepting symbol on stack we move to the accepting q_4 state
- q_4 - Accepting state, there is no moving out of this state since we checked that the input is already empty
- q_5 - This state is accessible if we went down case 2 route, we keep on pushing 1 from input to stack and once we receive 2 we go to the q_6 state
- q_6 - This state is accessible if we went down case 2 route, we keep on *popping* 1 from stack if we receive 2 and once the input is empty **AND** and we have accepting symbol on top of the stack we move to the accepting q_4 state.

Chapter 4

Transition functions

Write the transition function (as triplets...).

$$\delta(q_0, 0, Z) = (q_0, 0Z)$$

$$\delta(q_0, 0, 0) = (q_0, 00)$$
$$\delta(q_0, 1, 0) = (q_1, 10)$$
$$\delta(q_0, 3, 0) = (q_3, \epsilon)$$
$$\delta(q_0, 1, Z) = (q_4, Z)$$
$$\delta(q_1, 1, 1) = (q_1, 11)$$
$$\delta(q_1, 2, 1) = (q_2, \epsilon)$$
$$\delta(q_2, 2, 1) = (q_2, \epsilon)$$
$$\delta(q_2, 3, 0) = (q_3, \epsilon)$$
$$\delta(q_3, 3, 0) = (q_3, \epsilon)$$
$$\delta(q_3, \epsilon, Z) = (q_4, Z)$$
$$\delta(q_5, 1, 1) = (q_5, 11)$$
$$\delta(q_5, 2, 1) = (q_6, \epsilon)$$
$$\delta(q_6, 2, 1) = (q_6, \epsilon)$$
$$\delta(q_6, \epsilon, Z) = (q_4, Z)$$

Chapter 5

Instantaneous description accepting example

Using instantaneous descriptions show how this PDA accepts a sequence belonging to the language.

Chapter 6

Instantaneous description rejecting example

Use instantaneous descriptions to show the behavior of PDA for input not belonging to the language.