Joshua Reed

915743782

Chapter 3

Question 5

Consider the language AⁿBⁿCⁿ = {aⁿbⁿcⁿ : n ≥ 0}, discussed in Section 3.3.3. We might consider the following design for a PDA to accept AⁿBⁿCⁿ: As each 'a' is read, push two a’s onto the stack. Then pop one a for each 'b' and one a for each c. If there is nothing left on the stack after all the input has been read, accept Otherwise reject.

Why does approach fail?

This approach fails to take into account that there could be half as many A’s as there are B’s and C’s combined and would still succeed under this FSM. Imagine if you had the string aabbbc.

This string however is not in its correct form, another example of a string that would get accepted is aaabbbbbc. This FSM accepts string of the form of:

)

Chapter 4

Question 2a.

Given a Java program p and the input 0, consider the question, “Does p ever output anything?” Describe a semidecision procedure that answers this question.

The program has only two different decisions when given the input of 0. When the Java program is given the input of zero it either never outputs anything and runs forever or halts.

Chapter 5

Question 2d.

Create a state diagram for the FSM that accepts L = {w ∈ {0, 1}\* : w corresponds to the binary encoding, without leading 0’s, of natural numbers that are powers of 4}.

The graph I created includes the dead state that could occur. I also included two final states seeing as 1 in binary is included in the possible powers of four. The general idea is that every representation of a power of four has a 2n multiple of preceding zeros, ex. 100,10000,1000000.

