Rajat Sethi – CPSC 3500 – Assignment 4

Question 1:

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| * A | BC | none |
| B | none | D |
| C | none | E |
| D | BC | none |
| E | F | none |
| F | BC | None |

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| * A | BC | None |
| BC | none | DE |
| DE | BCF | None |
| BCF | BC | DE |
| none | none | none |

A screenshot of a cell phone

Description automatically generated with low confidence

2.) To concatenate two FAs without epsilon-transitions, I propose the following method.

* For every accept state of the first automaton, if the starting character of the **second** automaton:
  + Does not have a Kleene Star:
    - (Example: a, aa\*, ab\*, (a+b), (a+b)b\*…)
    - Make a transition with that first character to a new node, which will transform into the remainder of the second automata.
    - Remove the accept state from the first automata.
  + Has a Kleene Star:
    - (Example: a\*, a\*b, (a+b)\*bb\*, a\*b\*)
    - Make a transition of the first character back to itself (The first automaton’s previous accept state) and continue building the second automaton from there.
    - Remove the accept state from the first automaton.
  + Is either ε or ε\* and there is nothing after it.
    - Do nothing, keep the accept states as they are.

3a.)

* Imagine three languages with alphabet {a, b, c} named X, Y, Z.
  + Let X = {All strings with an even amount of a’s}
  + Let Y = {All strings with an even amount of b’s}
  + Let Z = {All strings with an even amount of c’s}
* To create a language where all of the strings have an even amount of a’s, b’s, and c’s, we can use an intersection to create the following language.
* To use product construction, we can use DeMorgan’s law to utilize unions.
* In which:
  + Let = {All strings with an odd amount of a’s}
  + Let  = {All strings with an odd amount of b’s}
  + Let = {All strings with an odd amount of c’s}

3b.) Each one of these FAs (X, Y, Z) and their complements can be made with two states (See sample figure for X). According to the rules of product construction, the amount of states in the resulting FA will be product of the amount of states in each individual FA.

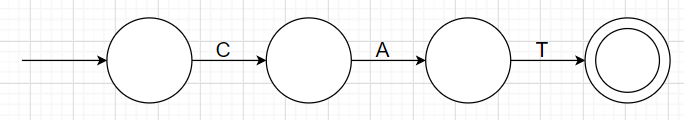
In this case, we get (2 states \* 2 states \* 2 states) = 8 states minimum.

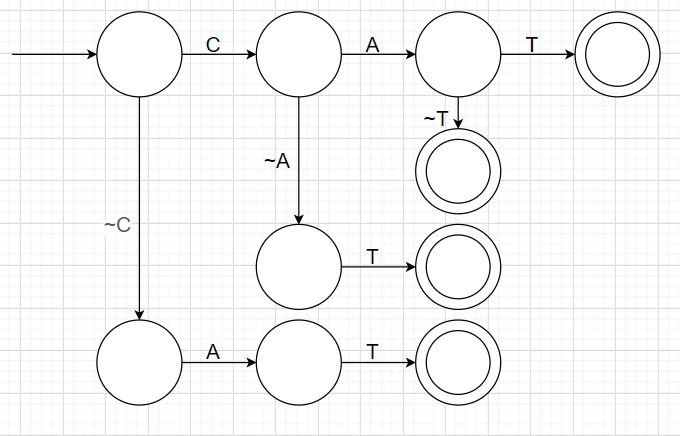
Icon

Description automatically generated with medium confidence

Finite Automata of Language X

4.) For every state in the original FA, create one extra transition that has the complement of the expected character pointing to a separate node. After that, recreate the remainder of the string without any possible mistakes.

Example: Let’s say I have the alphabet {CAT}. The FA would look like this:

By adding transitions that allow for ONE mistake, we can create an extended FA with several paths pointing to new accept states (Note: “~” refers to a complement).