Joshua Reed

CJ William Hong

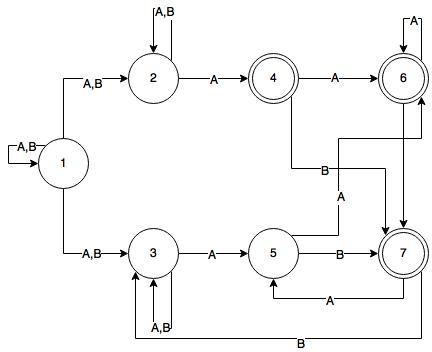
915743782, 913274185

Group Projects #3

1. L = {w ∈ {a, b}\* : |w| ≥ 2 ∧ the next-to-last symbol in w is a }

1.i

We began first by creating a NDFSM that could accept L. Beginning at state one the machine could either move to 2 or 3 and from 2 or 3 move to 4 or 5. The difference between 2,4,6 and 3,5,7 is that 2,4, and 6 are trying to accept a string with two a’s at the end. 3,5, and 7 are trying to accept a string that ends with ‘ab.’ We came to the conclusion that there were in total four different outcomes that mattered for this string. Those outcomes were a string that ended in ‘aa’,’ab’,’ba’, and ‘bb.’

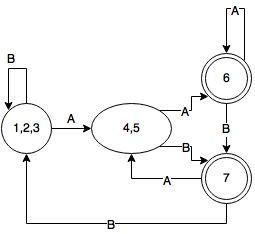


1.ii

We first created a state diagram to show the relationship between each state and their transitions.

|  |  |  |
| --- | --- | --- |
| K | δ | K’ |
| {1} | A | {1,2,3} |
|  | B | {1,2,3} |
| {1,2,3} | A | {1,2,3} & {4,5} |
|  | B | {1,2,3} |
| {4,5} | A | {6} |
|  | B | {7} |
| {6} | A | {6} |
|  | B | {7} |
| {7} | A | {5} |
|  | B | {3} |

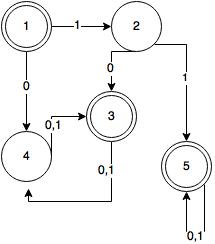
Our old states can now be combined into 4 new states, {1,2,3}, {4,5}, {6}, {7}. We can represent this DFSM as the following:



2. L = {w ∈ {0, 1}\* : every odd length string in L begins with 11}

2.i,ii

Equivalence classes describe the types of strings that each state encounters; The equivalence classes for this language represent null, odd, and even length strings and whether or not they have leading 11’s. You can construct an equivalence class table after first constructing the FSM. The FSM can be constructed as the following:



The idea is that a null string would get accepted due to it not being an odd string. The next state accepts a 1 but is not an accepting state due to there not being two 1’s at the beginning of the string. If state 3 is the next state to transition to then there will never be an accepting odd length string due because the beginning of the string is represented by 10, however; even strings always get accepted and that is why it is an accepting state. If the transition goes from 1 -> 2 -> 3, then every string gets accepted because there are two leading 1’s and every even string gets accepted.

The equivalence classes for these states can then represented as:

|  |  |
| --- | --- |
| 1 | [ε] |
| 2 | [1] |
| 3 | [Even length strings] |
| 4 | [Odd length string without two leading 1’s] |
| 5 | [11 + {0,1}\*] |