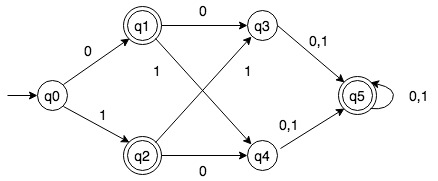
Solutions of Assignment #1 --- CSC320, Summer, 2018

Zhaocheng Li

V00832770

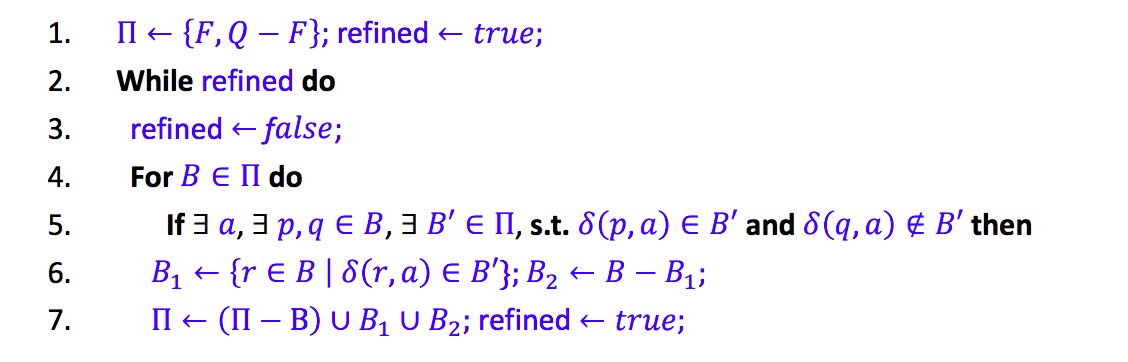
Q#1-(a):

recall the given graph,



With the instructions given in class, here we are going to trace the procedure to the solution, which is the minimal automaton.

The procedure is given as follow:



Trace:

1. F = {q1, q2, q5}, Q-F = {q0, q3, q4}, a∈{0, 1}
2. Π = {F, Q-F}, refined = true

* refined = false
* B = F, B’=Q-F

1. a = 0, δ(q1, 0) = q3 ∈ B’; δ(q2, 0) = q4 ∈ B’; δ(q5, 0) = q5 ∉ B’
2. B1 = {q1, q2}, B2 = {q5}
3. Π = {{q1, q2}, {q5}, {q0, q3, q4}}, refined = true

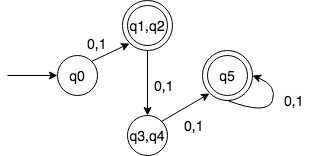
* B = Q-F, B’ = {q5}

1. a = 0, δ(q0, 0) = q1 ∉ B’; δ(q3, 0) = q5 ∈ B’; δ(q4, 0) = q5 ∈ B’
2. B1 = {q3, q4}, B2 = {q0}
3. Π = {{q1, q2}, {q5}, {q3, q4}, {q0}}, refined = true
4. refined = false
5. PROGRAM TERMINATED

After the partitioning, we have Π = {{q1, q2}, {q5}, {q3, q4}, {q0}} as final collection.

By constructing the minimal automaton M’= (Q’, ∑, δ’, [q0], F’) where

* Q’ = {{q1, q2}, {q5}, {q3, q4}, {q0}}
* F’ = {{q1, q2}, {q5}}
* δ’([q], a) = [δ(q, a)]
* [q0] = {q0}

Hence, we have constructed such minimal DFA, visualizing as follow

Q#1-(b):

given the alphabet ∑ = {0,1}, and the minimized DFA. Then the language accepted by this automaton is:

{w | w∈Q and |w|=1 or |w| > 2}

Q#2-(a)

Proof: given L = {w = 0n1m0n | m, n ≥ 0}, we give the proof by pumping lemma.

Take p as the “pumping length”, and consider the string w0=0p1q0p, such that |w0| ≥ p, we can break down w0 into three strings, w0=xyz, satisfying the conditions:

1. |xy| <= p
2. y ≠ ε
3. xykz ∈ L for each k ≥ 0

Now since |xy| <= p, it must be the case that xy = 0i for some i ≥ 0. Since y≠ ε, it must be the case that y=0j with j > 0. Taking k = 0, so

xykz = xy0z = 0p-j1q0p ∉ L, since there are more 0’s behind 1’s than 0’s before 1’s. Therefore, L is not regular.

Q#2-(b)

Proof: Given L = {0m1n | m ≠ n}. Here we can prove by contradiction. Assuming that L is regular, and by the closure properties of regular languages. Let M be the complement of L and hence, M is regular.

Since M = {0m1n | m=n}, consider the string w = 0p1p, with pumping lemma p. Then |w| = 2p > p, we can break down w into three strings, w=xyz, satisfying the conditions:

1. |xy| <= p
2. y ≠ ε
3. xykz ∈ L for each k ≥ 0

Now since |xy| <= p, it must be the case that xy = 0i for some i ≥ 0. Since y≠ ε, it must be the case that y=0j with j > 0. Taking k = 0, so

xykz = xy0z = 0p-j1p ∉ L, the numbers of 0’s and 1’s are not equal, contradicting the assumption that M is regular.

So, L is not regular.

Q#2-(c)

Proof: Given L = {wtw | w, t ∈ {0, 1}+}

Consider wtw = 0p10p, with pumping lemma p. such that |wtw| ≥ p, we can break down wtw into three strings, wtw=xyz, satisfying the conditions:

1. |xy| <= p
2. y ≠ ε
3. xykz ∈ L for each k ≥ 0

Now since |xy| <= p, it must be the case that xy = 0i for some i ≥ 0. Since y≠ ε, it must be the case that y=0j with j > 0. Taking k = 0, so

xykz = xy0z = 0p-j10p ∉ L, since there are more 0’s behind 1’s than 0’s before 1’s. So, L is not regular.

Q#3-(a):

given the language L = {w ∈ {0, 1}\* | w = wR }, the CFG could be the set G = ( {S}, {0, 1}, P, S), and the set of P of productions would be:

* S → ε
* S → 0
* S → 1
* S → 0S0
* S → 1S1

Q#3-(b):

given the language L = {w ∈ {0, 1}\* | w contains the same number of 0’s and 1’s}, the CFG could be the set G = ( {S}, {0, 1}, P, S), and the set of P of productions would be:

* S → ε
* S → 1S0
* S → 0S1

Q#3-(c):

given the language L = {w ∈ {0, 1}\* | w = 0n1n, n ≥ 0}, the CFG could be the set G = ( {S}, {0, 1}, P, S), and the set of P of productions would be:

* S → ε
* S → 0S1

Q#4:

Given the language A = {aibjck | i = j or j = k where i, j, k ≥ 0}. We have the corresponding CFG G=({A, B, C, D, E}, {a, b, c}, P, S), and the set of P of productions would be:

* A → B
* A → C
* B → aB
* B → D
* C → Cc
* C → E
* D → ε
* D → bDc
* E → ε
* E → aEb

This CFG is ambiguous because we can find two distinct leftmost derivations, one is

* A ⇒ C ⇒ Cc ⇒ aEbc ⇒ abc

And the other could be

* A ⇒ B ⇒ aB ⇒ abBc ⇒ abc

Q#5:

Given the grammar;

* E → E + T | T
* T → T \* F | F
* F → (E) | num

Into the CNF form:

* S0 → E
* E → EA
* A → RB
* B → RT
* T → TC
* C → WD
* D → WF
* F → num |YP |YZ
* P → EQ
* Q → EZ
* R → +
* W → \*
* Y → (
* Z → )

Q#6:

Given the CNF form for the grammar

E → E \* E | E + E | (E) | id | num

as:

* E → EA | EB | LD | id | num
* A → ME
* B → PE
* D → ER
* M → \*
* P → +
* L → (
* R → )

Consider CYK algorithm on w = ( id + num ) \* num

|w| = 7

We have a table T that is 7X7 (UPPER TRIANGLE ONLY) and V = { S0, E, A, B, D, M, P, L, R}, if we have s0 as the start variable

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L | LE | LEP | LEP | LPER | LPERM | LPERM |
|  | E | EP | EP | PER | PERM | PERM |
|  |  | P | PE | PER | PERM | PERM |
|  |  |  | E | ER | ERM | RME |
|  |  |  |  | R | RM | RME |
|  |  |  |  |  | M | ME |
|  |  |  |  |  |  | E |

Q#7: ALL CNF form are unambiguous, since

1. Unique new start variable
2. There is no more null terminal, so the order of each terminal in the fixed direction of derivation is fixed.
3. Each variable is assigned either one terminal or two variables, along each way you either reach to the end or further without loop back and termination.