CS340: Assignment 2

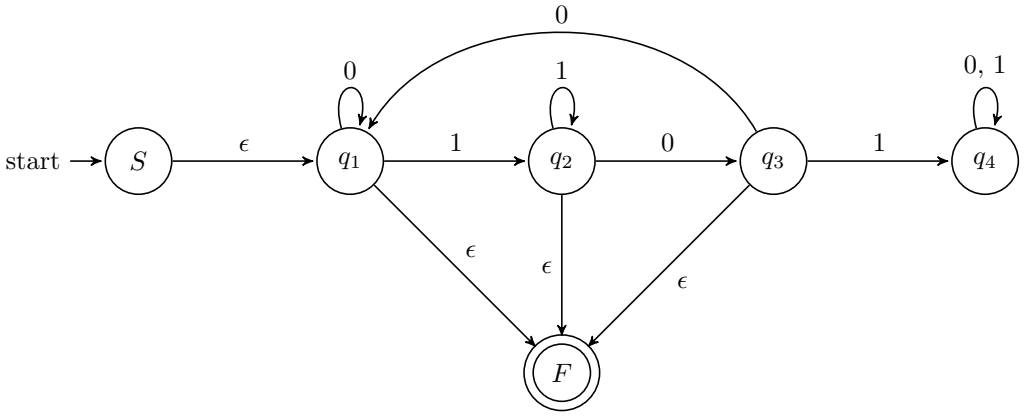
Gurpreet Singh . 150259

Question 1

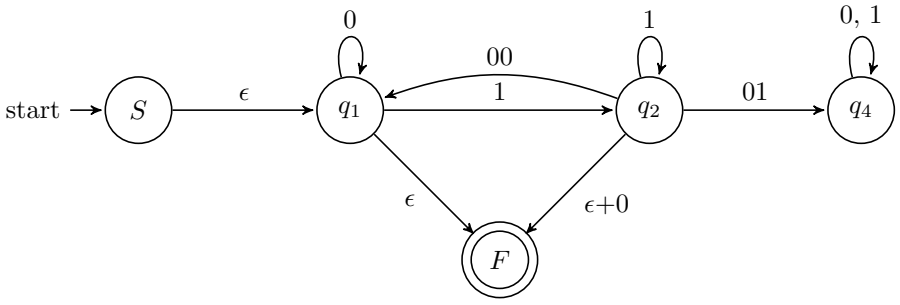
In order to find a regular expression, we can find a GNFA for the language, and reduce the GNFA one state at a time, in order to achieve a direct transition from the start state to the end state, thus obtaining the required regular expression.
We are given the language

$$L = \left\{x \in \{0,1\}\right\}^* | \text{ x does not contain the substring 101}$$

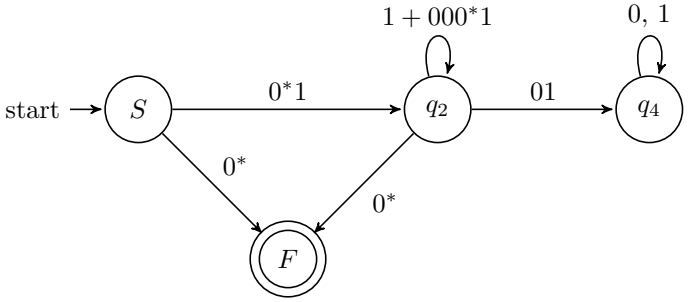
The simplest GNFA construction for this is



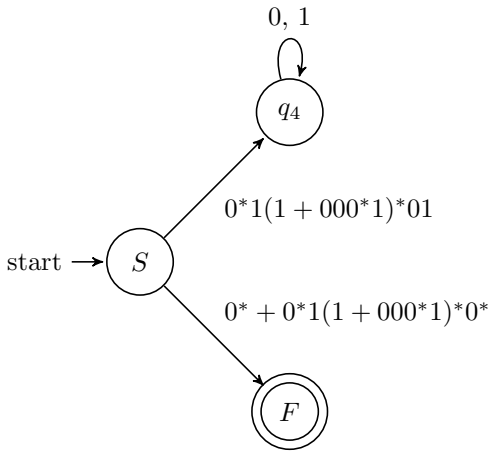
We can ignoring q_4 as it is a dump state. Therefore, we can reduce any of the three states q_i, $i \in [3]$, however we will reduce q_3 first, to achieve a simpler regular expression.



Further reducing by removing state q_1



Finally reducing by remove q_1 will give us



Hence, the regular expression that forms the language L is given by the regular expression

$$R = 0^* + 0^*1(1 + 000^*1)^*0^*$$

With some manipulation, we can further reduce this regular expression the the following

$$R = \mathbf{0^*(000^* + 1)^*0^*}$$

Question 2

We have the language

$$L = \left\{\{0^i1^j\} \,|\, gcd(i,j) = 1\right\}$$

In order to prove that L is not regular, we use the Pumping Lemma. Therefore, we first assume that L is regular. Hence, we can assume that there is a DFA D, such that $L(D) = L$. Let

$$D = D(Q, \Sigma, q_0, \delta)$$

Let $p = |Q|$. Consider a string $w = 0^p1^q$ such that q is a prime $\geq p$. Clearly, w belongs in the language L, and $|w| > p$.

We can partition this string $w = xyz$ according to the pumping lemma, such that $\forall\, i \in \mathbb{W}$, $w^i = xy^iz \in L$.

It is clear to see that we can write $y = 0^t$ where $t \leq p$. Now we know q is a prime, hence $gcd(t, q) = 1$.

Claim: For all $t < q$ (t and q as defined above), we can find $k \in \mathbb{W}$ such that $p + kt \equiv 0 \pmod{q}$
Proof:

Consider all values of k from q to $2q - 1$. Let us say, that for no values of $k \in [q, 2q-1]$, $p + kt \equiv 0 \pmod{q}$.

Since there are q values of k, by **Pigeon Hole Principle**, we can say that there is at least one value $r < q$, s.t two values of k give the same result modulo q. i.e.

$$\exists\, k_1, k_2 \in [q, 2q-1] \text{ s.t.} \quad p + k_1t \equiv p + k_2t \equiv r \pmod{q}$$

Subtracting these two, we get

$$(k_1 - k_2)t \equiv 0 \mod q$$

However, since $|k_1 - k_2|, t < q$, this is not possible (as q is prime). Hence, our initial assumption is false.

Using the above claim, we can say that we have found a value of $i = k + 1$, where $p + kt \equiv 0 \mod q$ which does not satisfy the pumping lemma. This suggests that the language L is non-regular, and our assumption was untrue.

Question 3

We have the language $L(A) \subset \{a\}^*$ such that $L(A) = \{a^m | m \in A\}$ where A is ultimately periodic.

If A is ultimately periodic, then we have $n, p \in \mathbb{N}$, such that $m \geq n$, $m + p \in A$ iff $m \in A$. This definition of A can be transformed into an alternate definition

$$A = A_0 \cup A_1 \cup A_2 \cdots \cup A_p$$

where A_0 is the set of numbers in $A < n$ and A_i is an AP which contains all the numbers in A which have *imodulop*. From the definition of A, we can say that A_i is either NULL or countably infinite.
We can also write L(A) in this form

$$L(A) = L(A_0) \cup L(A_1) \cup L(A_2) \cdots \cup L(A_p)$$

Claim: A_i is regular ($i \in [1, p]$)

Proof:

If A_i is NULL, then we are done, as this is regular. Rather, this does not affect the result at all.

If A_i is not NULL, then we can construct a DFA $D_i = (Q_i, \Sigma_i, q_i^0, F_i)$ such that $L(D_i) = L(A_i)$.

Let $a_i = \min(A_i)$. Since n is finite from the definition of A, a_i also needs to be finite. Therefore, we can easily construct a DFA, with a_i linear states, and then a cycle of length p. An example is show below

Since we have a DFA construction for A_i, we can say that A_i is regular.

Since $L(A_i)$ is regular for $i \in [1, p]$, we can say $L(A_1) \cup L(A_2) \cdots \cup L(A_p)$ is regular.

Also, since the number of elements in A_0 is finite ($< n$), hence $L(A_0)$ is also regular. Therefore, we can say $L(A)$ is regular if A is ultimately periodic.

For the reverse claim, we use contrapositivity. Say that A is not ultimately periodic. Then, at least one of the A_i has to be a non A.P. Let such an i be $i = k$. Hence we say A_k is not periodic.

Claim: $L(A)$ is not periodic if for any $k \in [1, p]$, A_k is not periodic.

Proof:

Let there be a number x such that for all elements in $A_k > x$, the set is periodic. But if this was the case, then we can set $n = x$, and the problem would be reduced to a periodic case. Therefore, no finite x exists.

In order to track non-periodic elements, we need a DFA to cover all the string length. However since this never becomes periodic, we can say that there exists no finite DFA which accepts $L(A_k)$. Hence $L(A_k)$ is non-regular. This implies that $L(A)$ is not regular as it is a union of at least one non-regular language with other languages. Hence, our claim holds true.

Using both the claims, we can finally say that $L(A)$ is regular if and only if A is ultimately periodic.

Note: We are assuming that the values of n and p are known before hand.

Question 4

Part A

$$L_1 = \left\{a^i b^j c^k d^l \mid i, j, k, l \geq 1, i = l, j = k\right\}$$

We need to ensure that all the elements are present at least once, and we need to keep the string symmetric.

$$\begin{array}{lcll} S & \longrightarrow & aSd & | \quad aS_1d \\ S_1 & \longrightarrow & bS_1c & | \quad bc \end{array}$$

Part B

$$L_2 = \left\{a^n b^m \mid n, m \geq 0, n \neq m\right\}$$

For this language, we divide the language into two cases $n > m$ and $n < m$.

$$\begin{array}{lcll} S & \longrightarrow & S_1 & | \quad S_2 \end{array}$$

$$\begin{array}{lcll} S_1 & \longrightarrow & aS_1b & | \quad A \\ S_2 & \longrightarrow & aS_2b & | \quad B \end{array}$$

$$\begin{array}{lcll} A & \longrightarrow & Aa & | \quad a \\ B & \longrightarrow & Bb & | \quad b \end{array}$$

Part C

$$L_2 = \left\{a^i b^j c^k \mid i, j, k \geq 0, i > j \text{ or } j > k\right\}$$

For this language, we divide the language into two cases $n > m$ and $n < m$.

$$
\begin{array}{lcll}
S & \longrightarrow & S_1 & | \quad S_2 \\
\\
S_1 & \longrightarrow & aS_1bC & | \quad Aa \\
S_2 & \longrightarrow & AbS_2c & | \quad Cc \\
\\
A & \longrightarrow & Aa & | \quad \epsilon \\
C & \longrightarrow & Cc & | \quad \epsilon
\end{array}
$$

Question 5

Part A
We can take $w = 1 + 1 + 1$ as the string. Clearly and $|w| = 5$ and w is ambiguous with respect to the CFG G. We can prove this by tracing multiple parse trees for this string. (Shown in the second part).

Part B
We can see two parse trees in the figure below for the string $w = 1 + 1 + 1$. This proves that the string w is ambiguous with respect to G

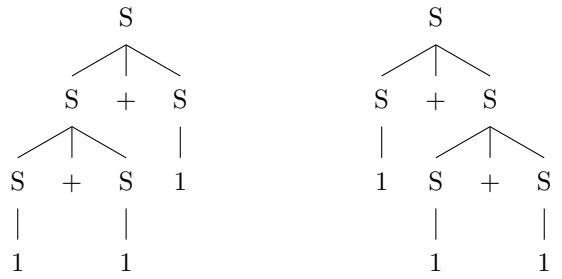


Figure 1: Two parse trees for the string $w = 1 + 1 + 1$

Part 3
The CFG which accepts the same language as G, and is ambiguous can be given by the following set of rules (say $\hat{G}$)

$$
\begin{array}{lcll}
S & \longrightarrow & S + S_1 & | \quad S_1 \\
\\
S_1 & \longrightarrow & S_1 * S_1 & | \quad S_2 \\
S_2 & \longrightarrow & 1 \quad | \quad 0 & | \quad (S)
\end{array}
$$

This language considers the precedence of operators as it first divides the string using all it's free addition symbols (which have the lowest precedence) and form maximal independent terms, which are then divide by the multiplication symbols which are further bounded by parenthesis if required. This is done recursively, and hence the precedence is automatically taken care of.

Note: Free in the following context means not bounded by parenthesis.

Claim: $L(G) = L(\hat{G})$

Proof By Induction:

For this, we need to prove that $w \in L(\hat{G})$ iff $w \in L(G)$. In order to prove this, we induct on the length of the string w. Let w be a string in $L(G)$ s.t. $|w| = l$.

Induction Hypothesis: For any string $w' \in L(G)$ where $|w'| < l$, $w' \in L(\hat{G})$ iff $w' \in L(G)$
Base Case: Consider the strings of length one, 0 and 1. It is trivial to see that both these strings exist in $L(G)$ as well as $L(\hat{G})$
Proof:

Case 1: There is a free '+' in w, i.e. this is not preceded by any other operator, and adds two independent terms.
In this case, $w = w_1 + w_2$. Also $|w_1|, |w_2| < |w| = l$. According to the construction of $\hat{G}$, , $w_1, w_2 \in \hat{G}$. Therefore, using the induction hypothesis on, w_1 and w_2, we can say that $w_1, w_2 \in L(G)$. From the $S \longrightarrow S + S$ rule, we can say that $w_1 + w_2$ must also be in $L(G)$. Also, if $w \in L(G)$, then with similar deconstruction and construction, we can say that $w \in L(\hat{G})$ if and only if $w \in L(G)$.

Case 2: w is of the form (w')
This case is trivial, as we can simply reduce w to w' and then construct back w since the rule $S \longrightarrow (S)$ exists in both $L(G)$ and $L(\hat{G})(indirectly)$. Hence the claim holds for this case

Case 3: There is no free '+' in w, and there is a '*'.
This case is similar to Case 1, and hence the induction hypothesis also holds for this case.

Hence our claim is true, and $L(G) \sim L(\hat{G})$

Claim: $L(G)$ is unambiguous

Proof By Induction:

We need to prove that for each string $w \in L(\hat{G})$ where $|w| = l$, there exists a unique left most derivation. Again, we induct on the length of the string w.
Induction Hypothesis: Let this be true for all strings with length less than l
Base Case: Consider the strings of length one, i.e. 1 and 0. There is clearly a unique derivation for both these strings. i.e. $S \longrightarrow S_1 \longrightarrow S_2 \longrightarrow 0 \quad | \quad 1$
Proof:

Consider a string w where $|w| = l$. We will try to reduce w to a smaller instance, in order to prove the claim. There are three cases for w

Case 1: w contains a free '+'
Since in this case, we cannot substitute $S \longrightarrow S_1$ as the first transition, since S_1 contains no free '+'. Hence, there is only one (left most) transition possible to reach w from the parent node (in the parse tree), i.e. $S \longrightarrow S + S_1$. Hence, we can divide the string w as $w = w_1 + w_2$. Since w_1 and w_2 are smaller than w, they follow the induction hypothesis. Hence, we know that there is a unique parse tree for both w_1 and w_2, and therefore there is only a unique left most derivation to obtain w.

Case 2: w is of the form (w')
This case is trivial, as there is only one computation path to achieve this from the smaller instance w', i.e. $S \longrightarrow S_1 \longrightarrow S_2 \longrightarrow (S)$. There is no other way to reach w, hence through the induction hypothesis on w' as $|w'| < l$, we can say there is a unique left most derivation for w.
Case 3: w has no free '+' and is of the form $w_1 * w_2$. where the '*' is the left most possible multiplication symbol.
This is similar to case 1, but the key here is that we are considering only the left most derivation, in which case, we will be reducing the parse tree of w to those of w_1 and w_2 where w_1 should contain no free multiplication symbol. With a similar approach as in case 1, we get a unique left most derivation for w.

Since we have proved for all cases, we can say that our claim, $L(\hat{G})$ is unambiguous is true.