Assignments 1 to 3

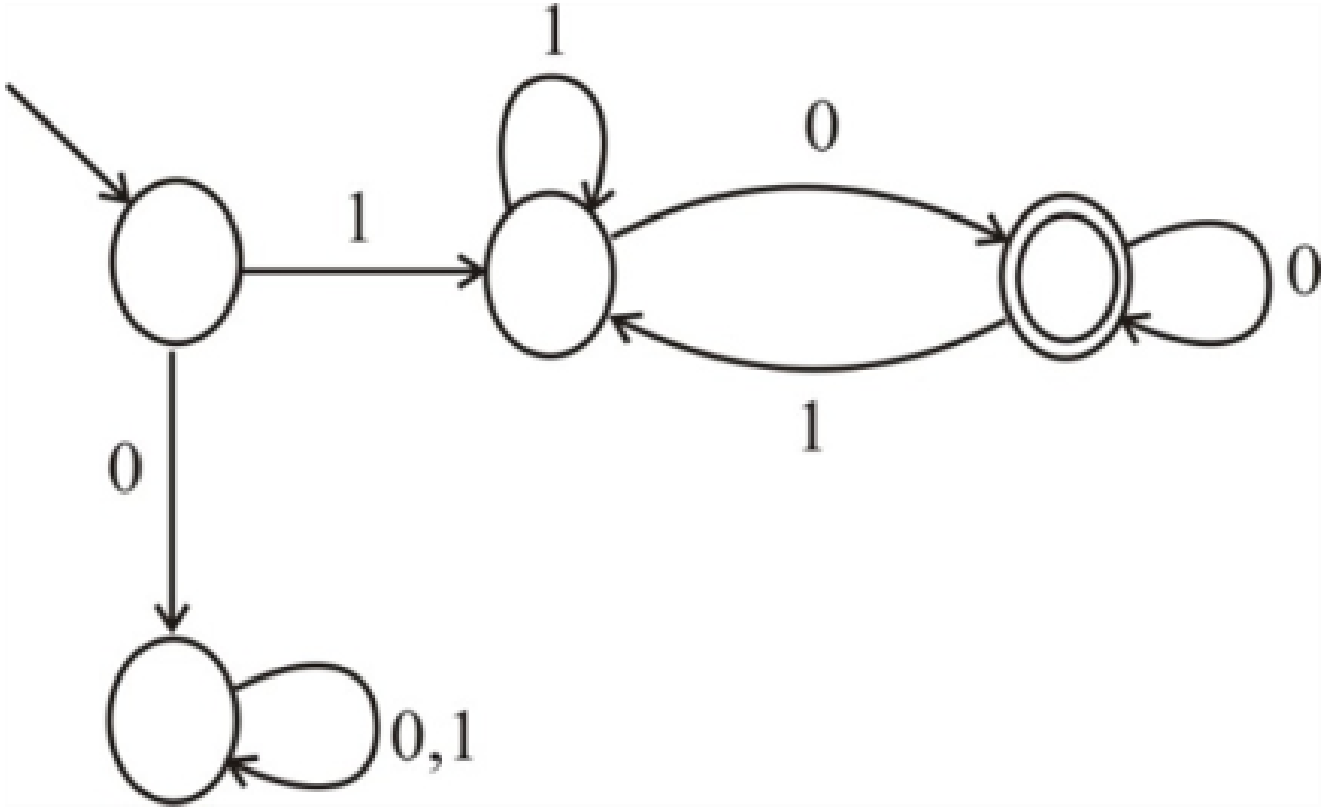
Regular Languages and Finite Automata

2022

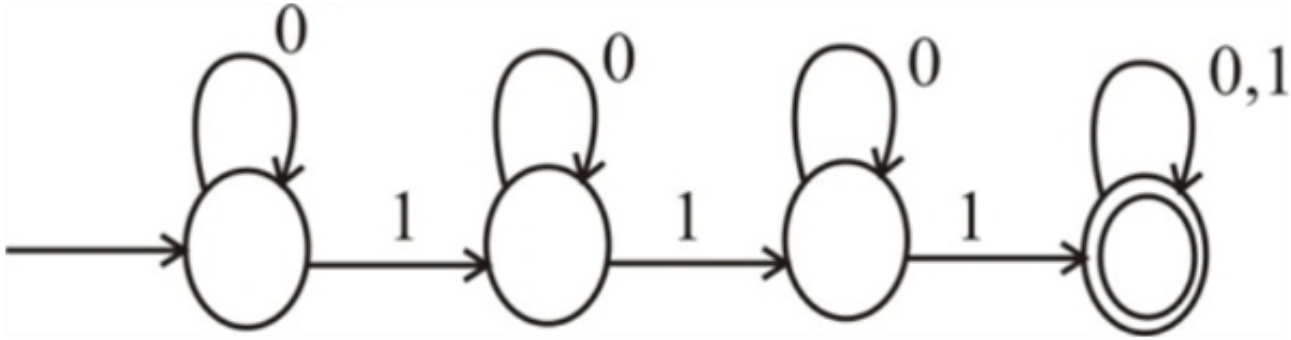
1-6

Give state diagrams of DFAs recognizing the following languages. In all parts, the alphabet is $\{0, 1\}$

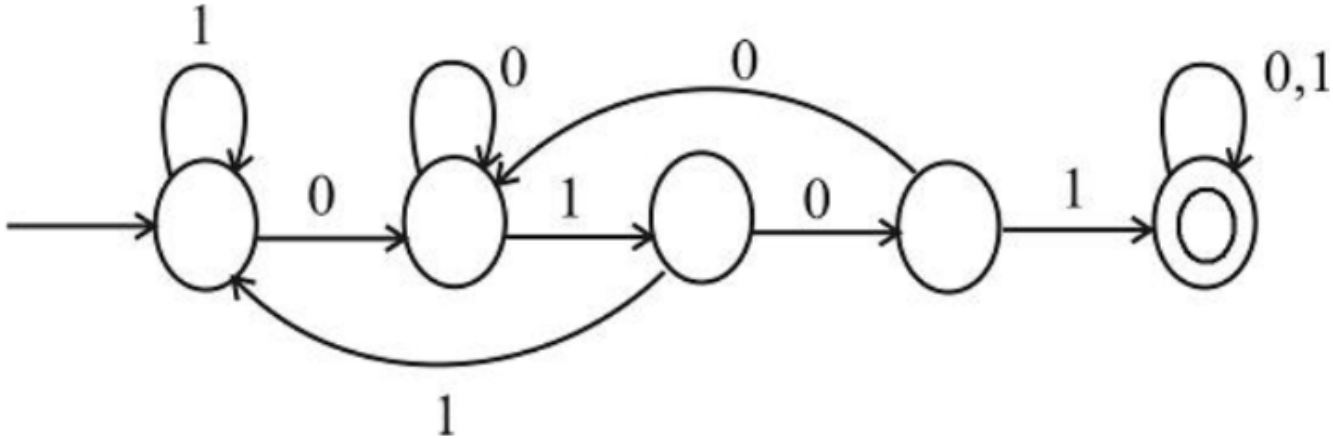
- {w|w begins with a 1 and ends with a 0}



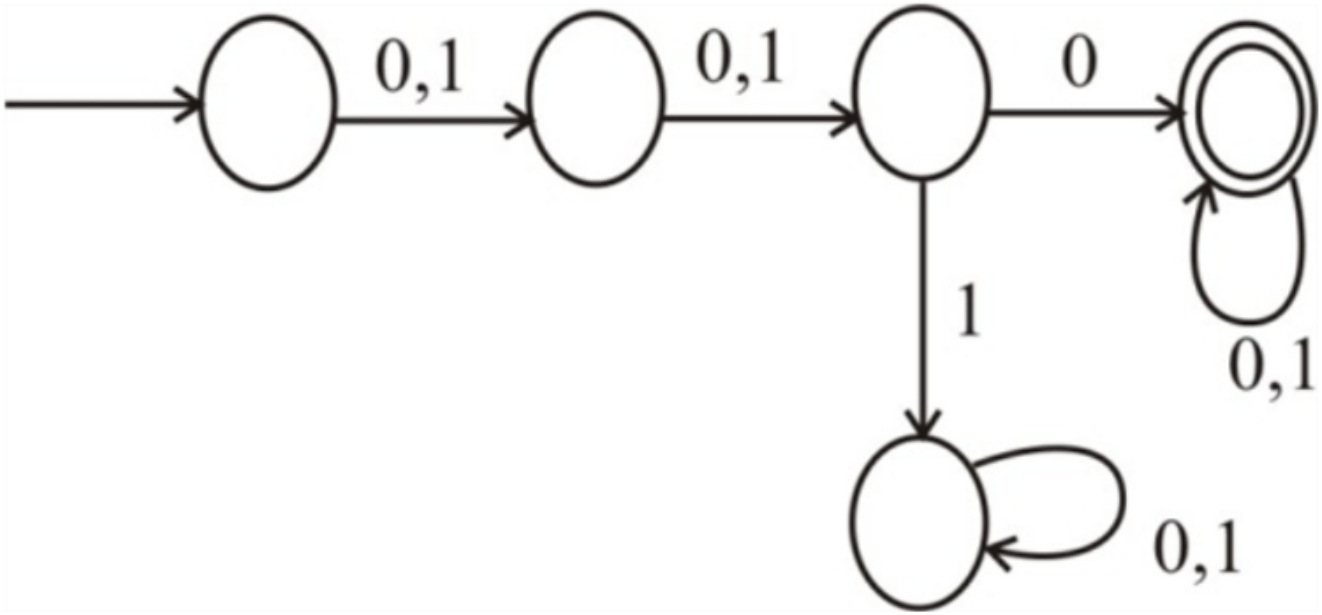
- {w|w contains at least three 1s}



- {w|w contains the substring 0101}



- {w|w has length at least 3 and its third symbol is a 0}



1-6

- {w|w starts with a 0 and has odd length, or starts with 1 and has even length}

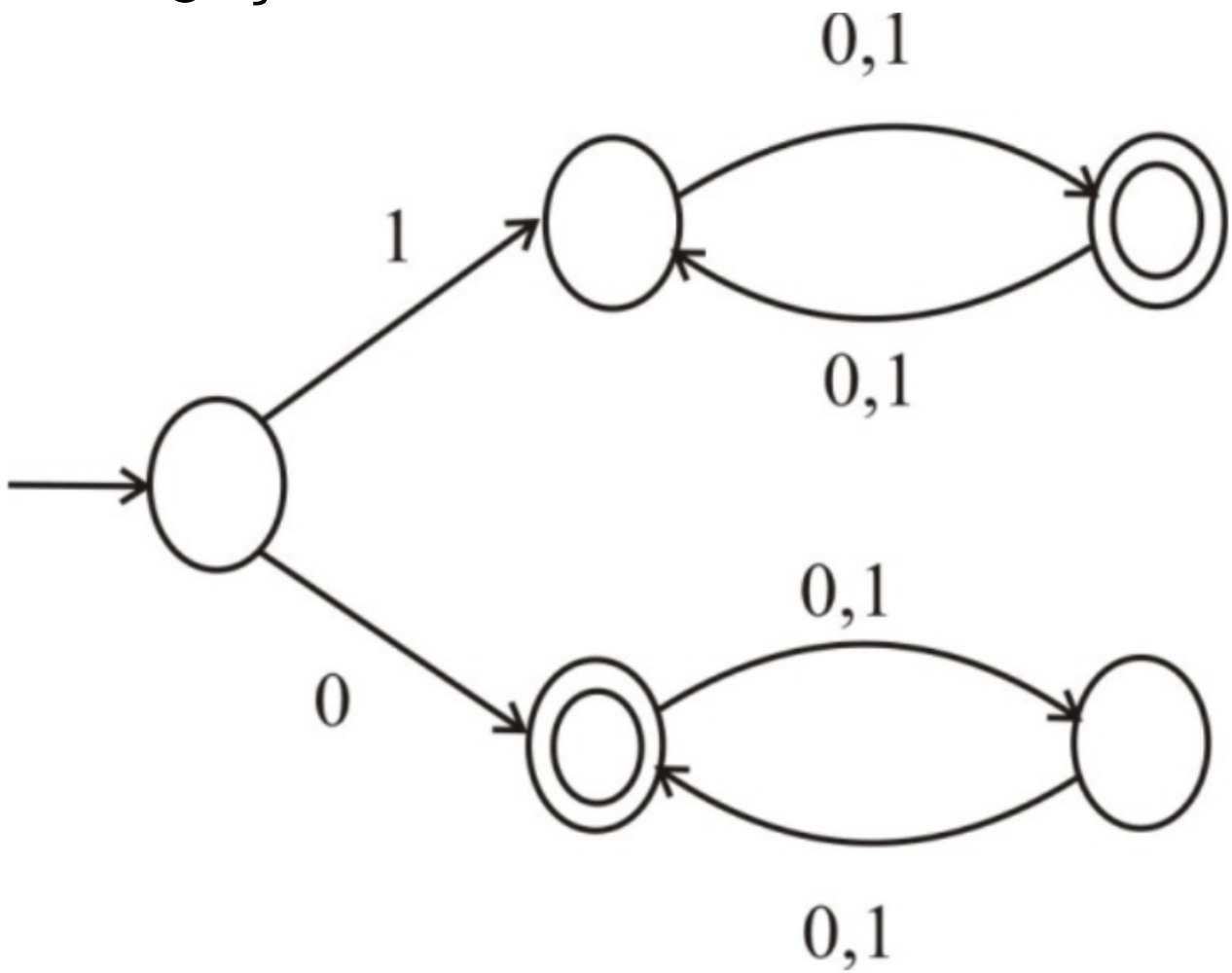
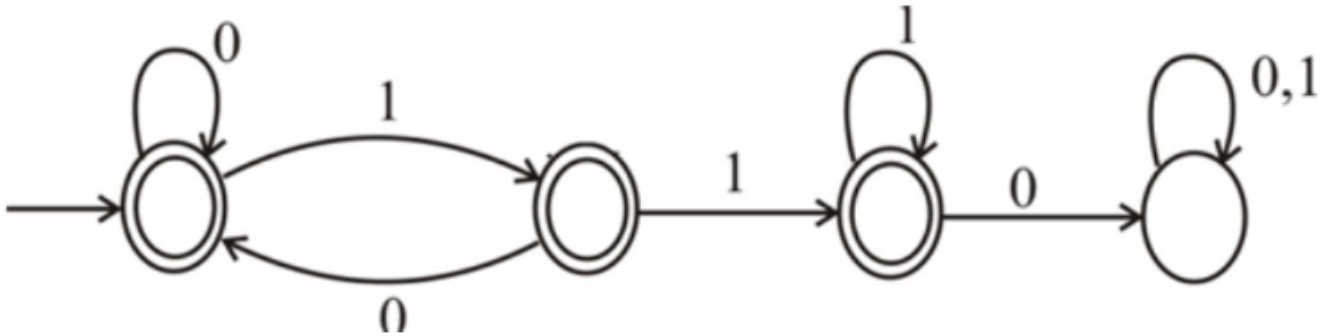
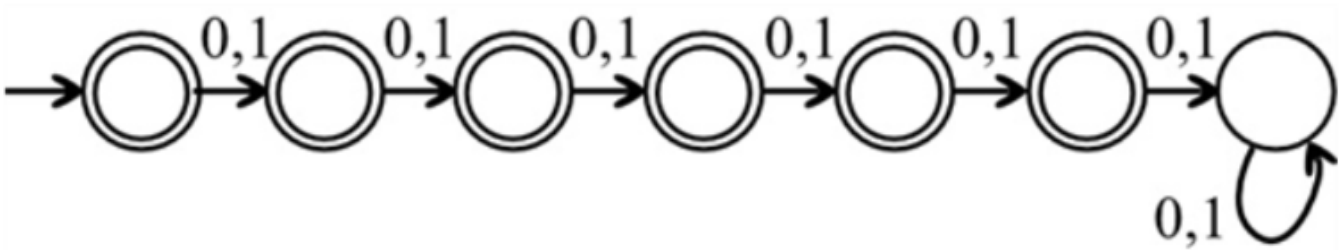


Figure 1:

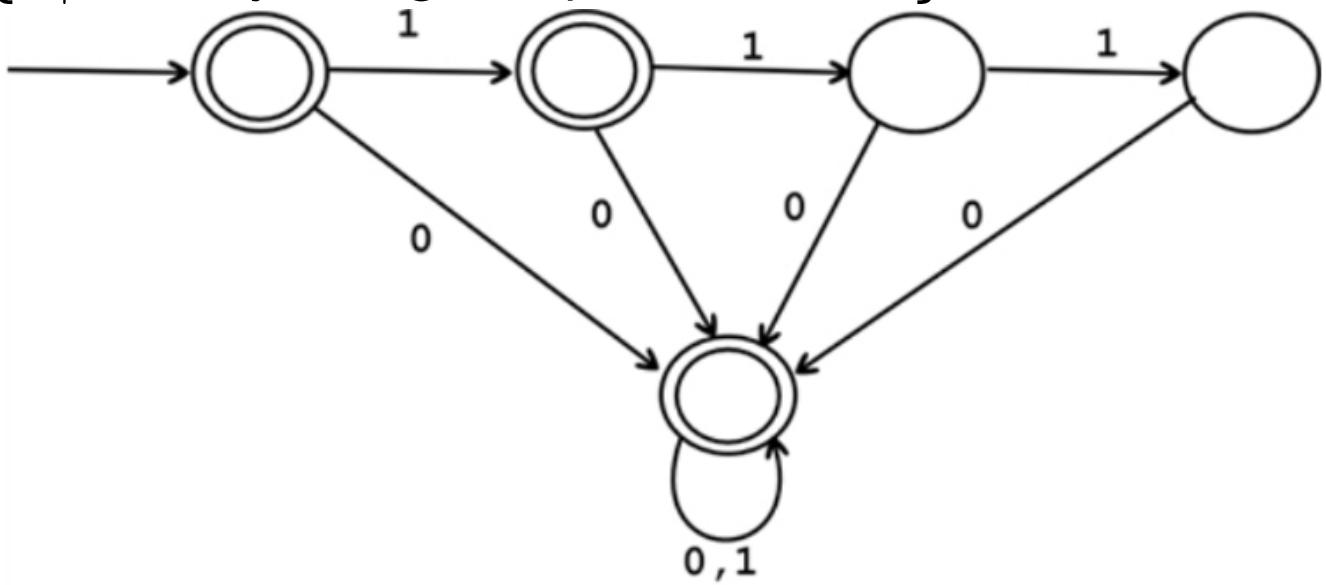
- {w|w doesn't contain the substring 110}



- {w| length of w is at most 5}

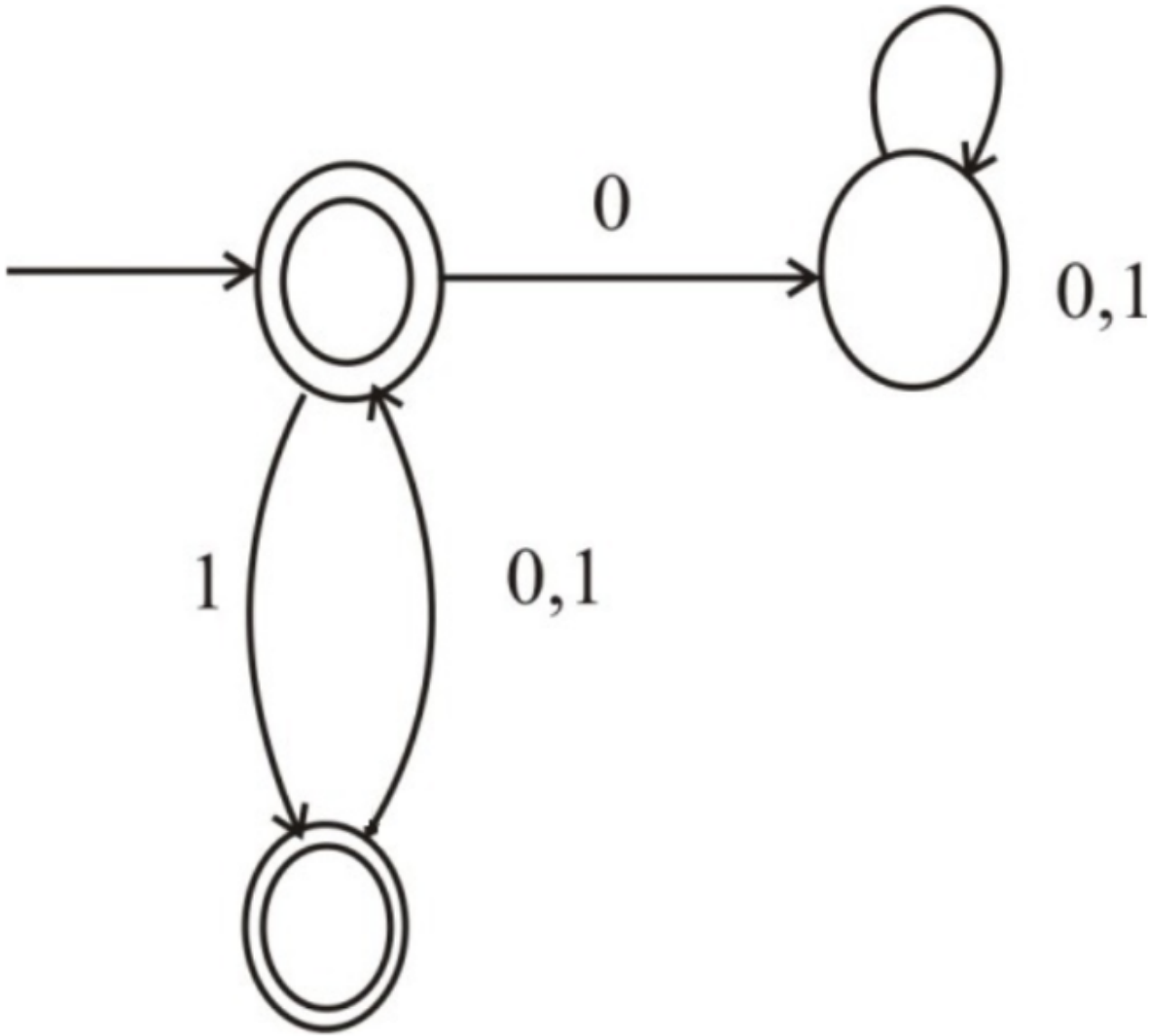


- {w|w is any string except 11 and 111}

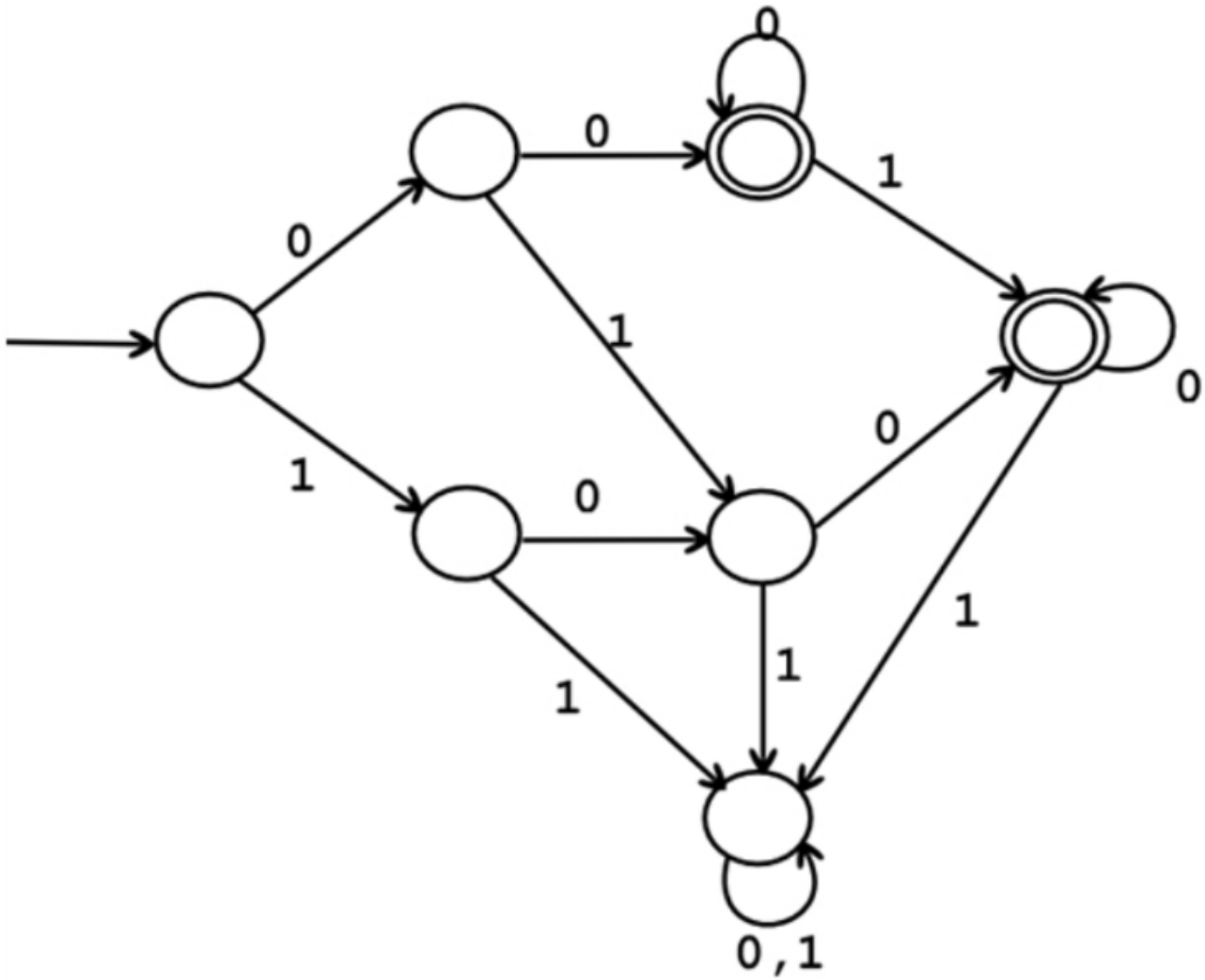


1-6

- {w|w every odd position of w is a 1}

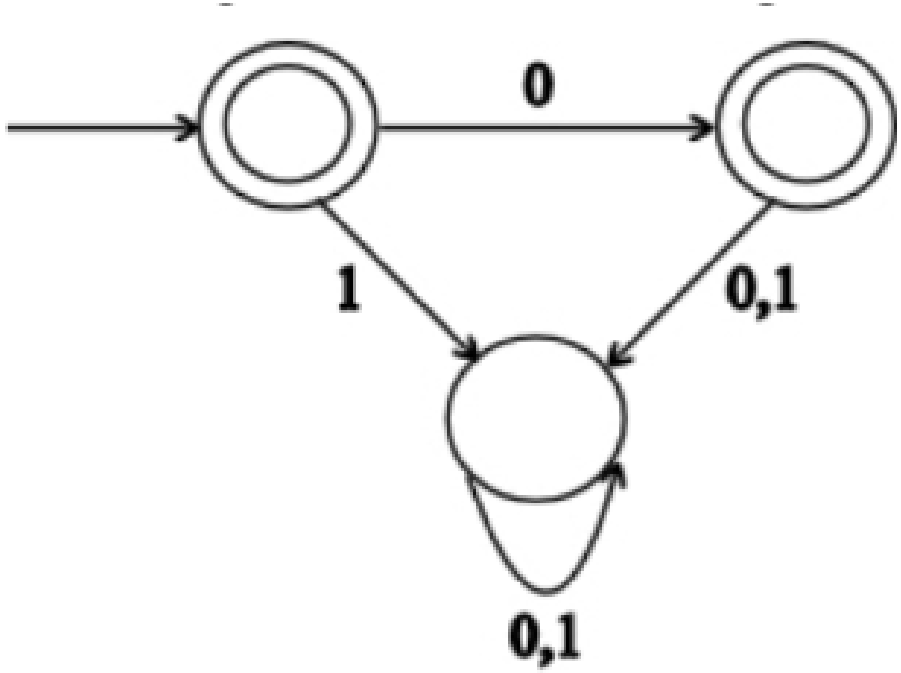


- {w|w contain at least two 0s and at most one 1}



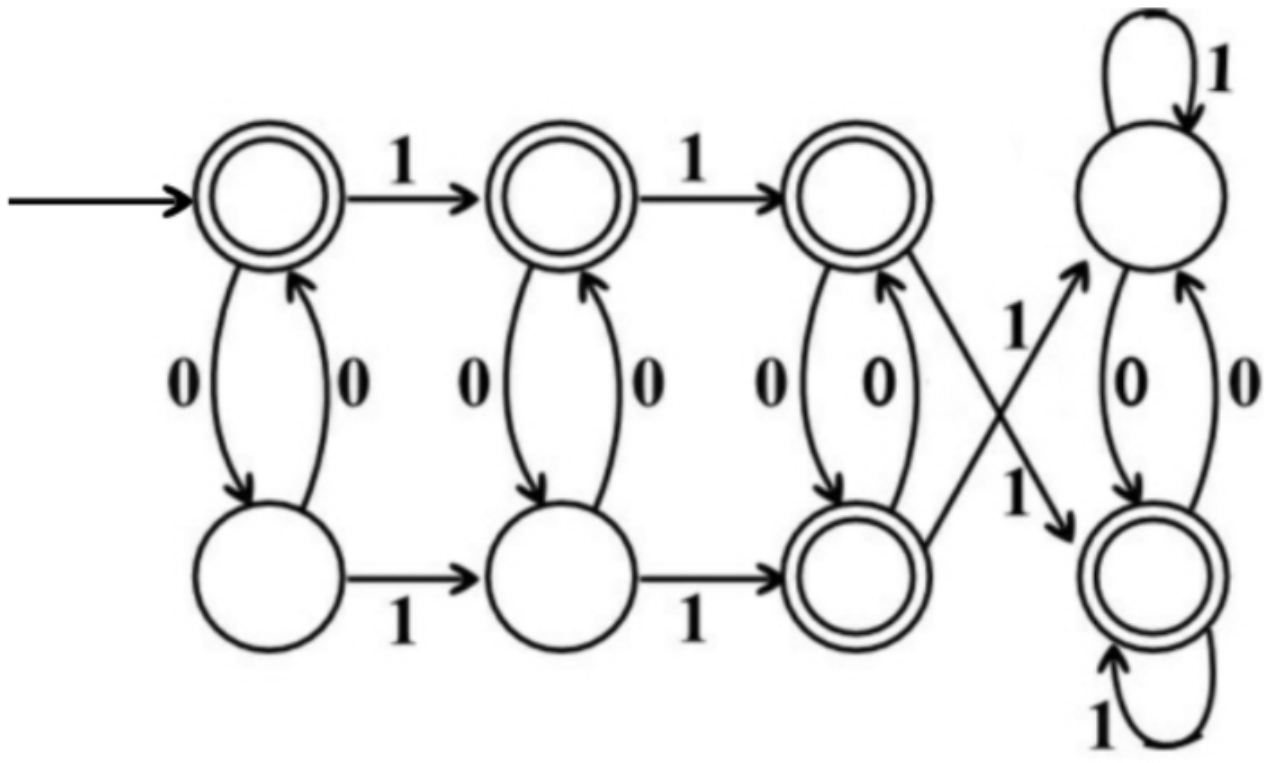
1-6

- $\{\epsilon, 0\}$

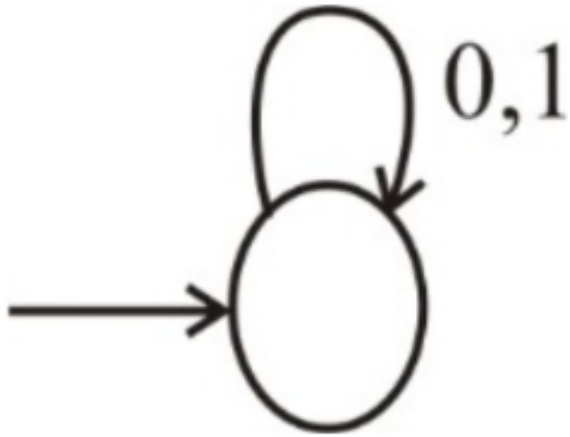


1-6

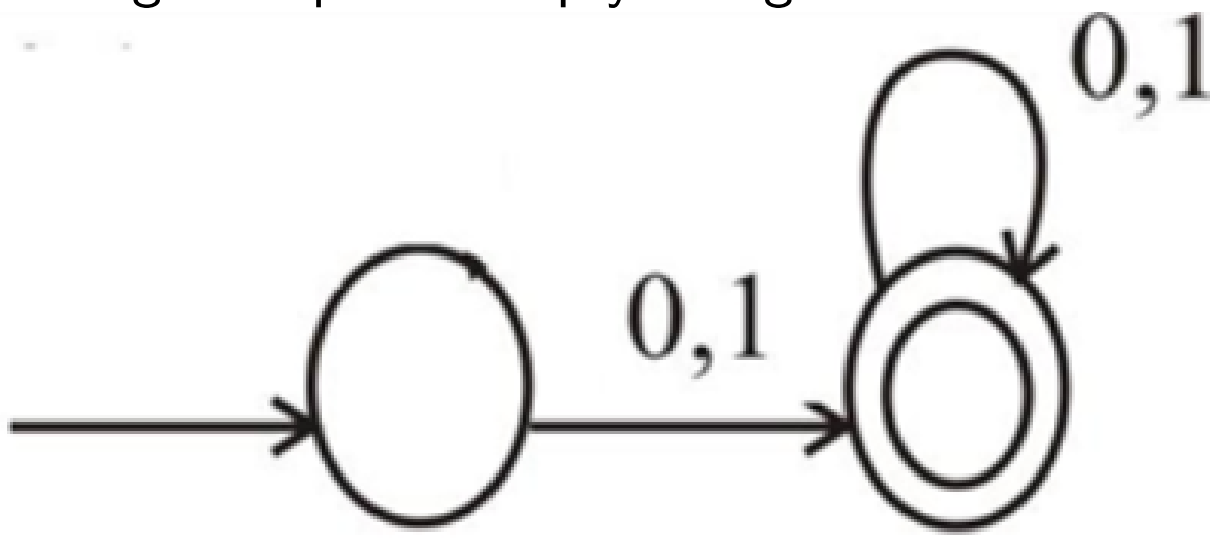
- {w|w contains an even number of 0s, or contains exactly two 1s}



- The empty set



- All strings except the empty string



1-11

Prove that every NFA can be converted to an equivalent one that has a single accept state.

1-11

Let M be a NFA. Let N be the another NFA with single accept state q_{final}. We go througn every accept state of M and do the following:

1. make it non-accepting state
2. add an ϵ-transition from that state to q_{final}

Then we will get NFA N. Note that if M has no accept state, then there will be no transitions coming into q_{final}

Now we discuss this formally:
$M = (Q, \Sigma, \delta, q_0, F)$ then $N = (Q \bigcup \{q_{final}\}, \Sigma, \delta', q_0, \{q_{final}\})$,
that for any $q \in Q$ and $a \in \Sigma$,
$\delta'(q, a) = \mathbb{1}_{a=\epsilon \wedge q \in F} \times \delta(q, a) \bigcup \{q_{final}\} + (1 - \mathbb{1}_{a=\epsilon \wedge q \in F}) \times \delta(q, a)$,
with $\delta'(q_{final}, a) = \emptyset$.
Thus we get N by simply making every accept state of M as non-accepting state and adding an ϵ-transition from that state to q_{final}, which means that M is equivalent to N. Thus every NFA is converted to an equivalent one that has single accept state.

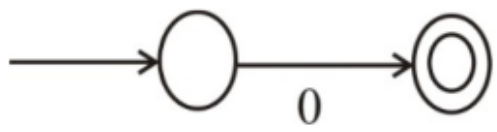
1-19

Use the procedure described in Lemma 1.55 to convert the following regular expressions to nondeterministic finite automata.

- $(0 \bigcup 1)^*000(0 \bigcup 1)^*$

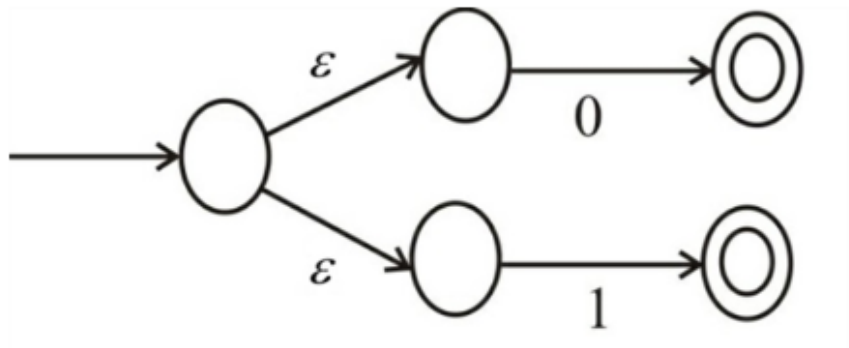
1-19

0

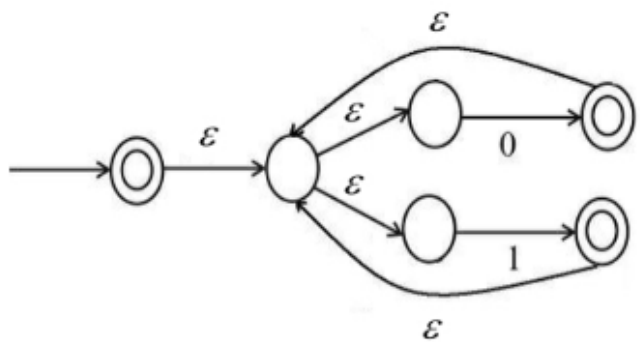


1

$0 \cup 1$

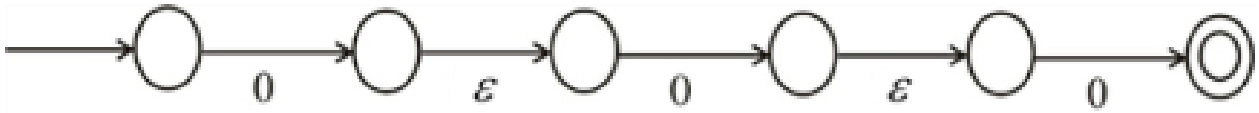


$(0 \cup 1)^*$

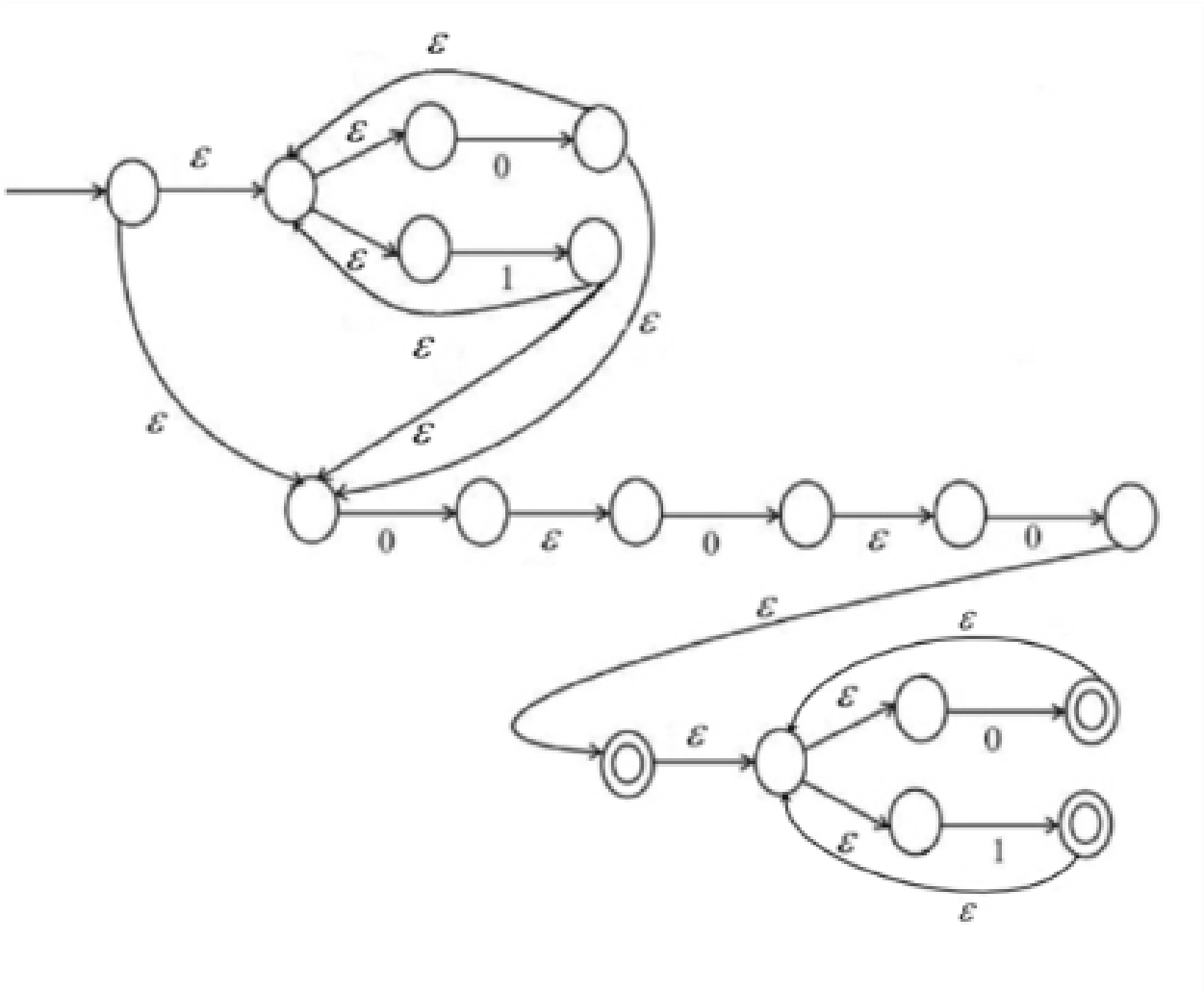


1-19

000



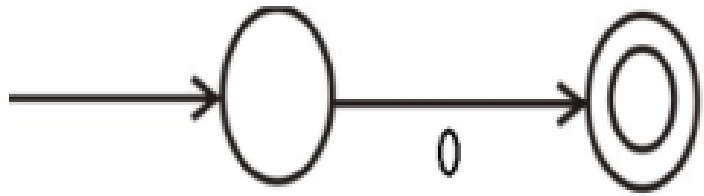
$(0 \cup 1)^*000(0 \cup 1)^*$



1-19

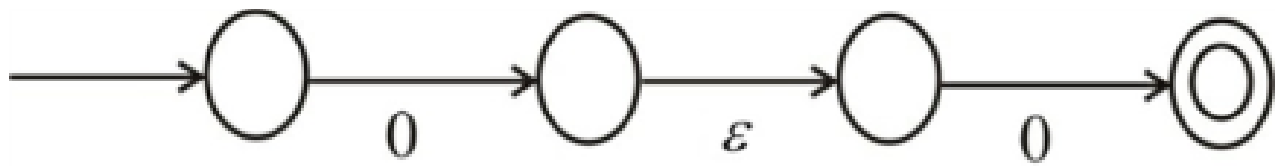
- $(((00)^*(11)) \bigcup 01)^*$

0

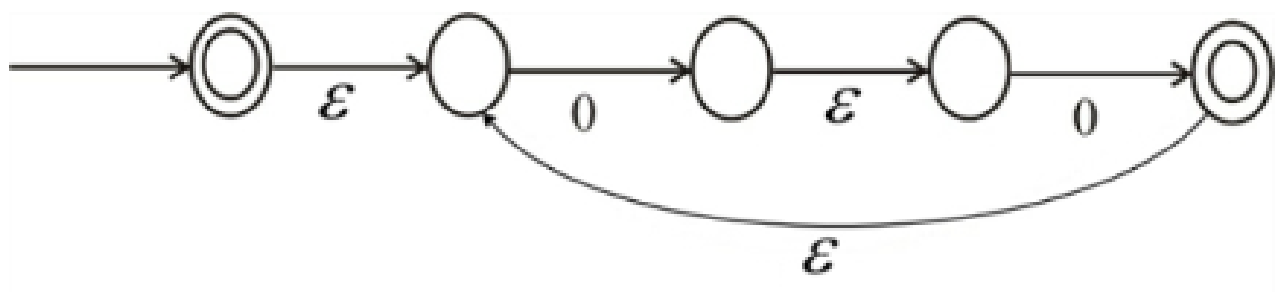


1

00



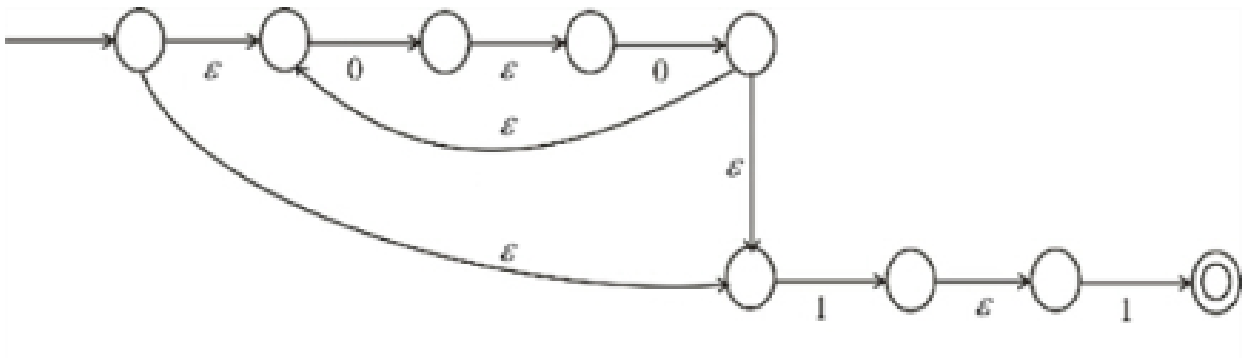
$(00)^*$



1-19

11

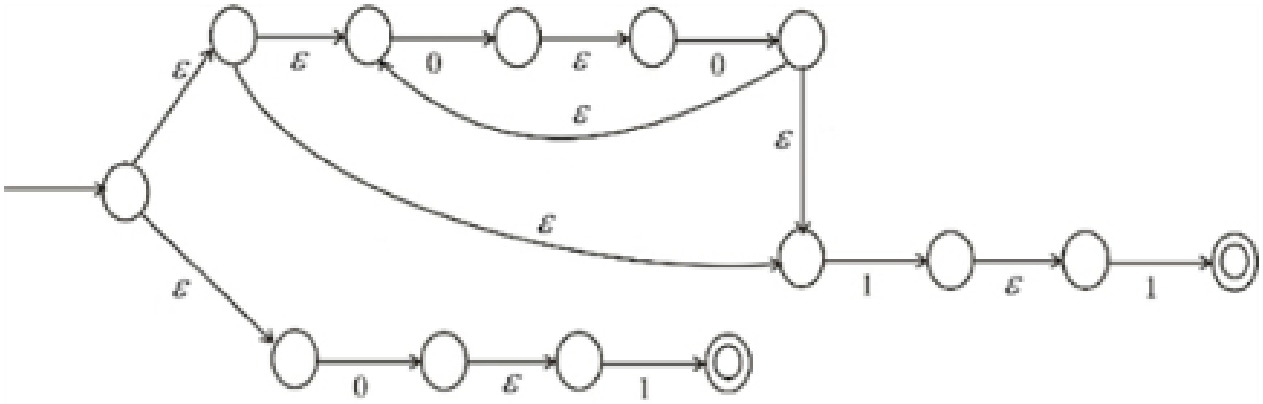
(00)*11



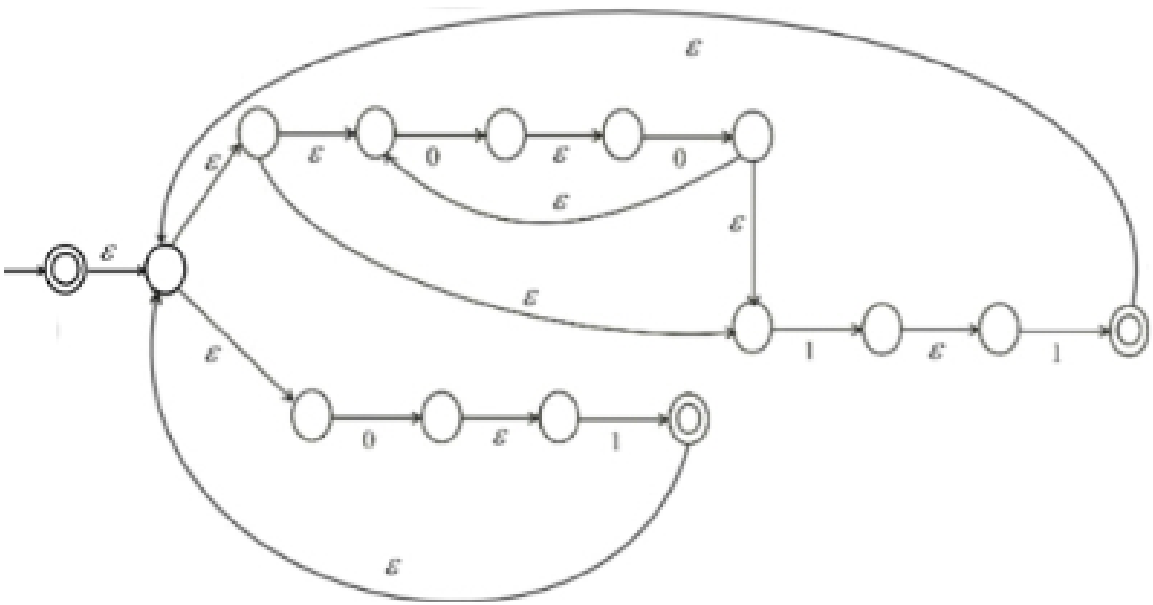
01

1-19

$(((00)^*(11))\cup(01))$



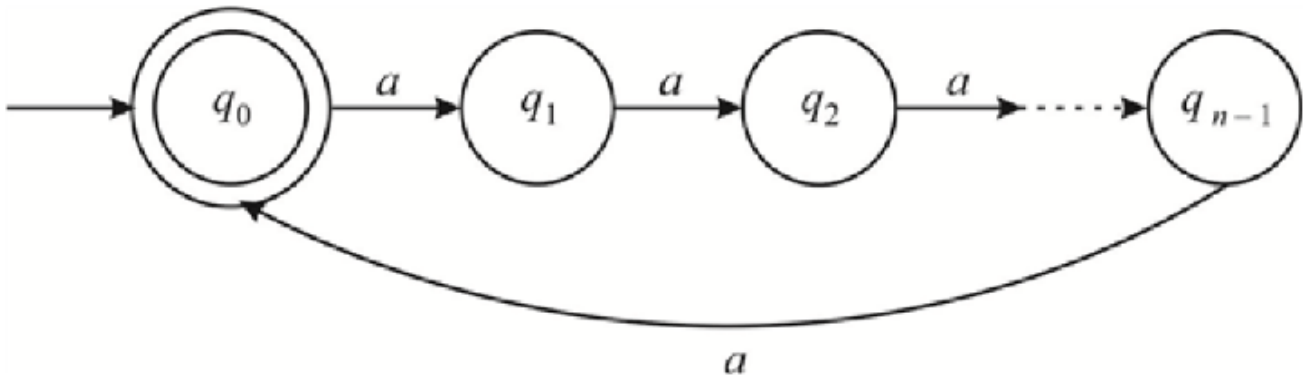
$(((00)^*(11))\cup(01))^*$



1-36

Let $B_n = \{a^k | k \text{ is a multiple of } n\}$. Show that for each $n \geq 1$, the language B_n is regular

Suppose that $k = ni$, using induction on i and a finite automaton as follow:



This proves that if $\{a^n\}$ is a regular language, then B_n is also regular.
After that, we can use the closure property of union to prove that all $\{a^n\}$ is regular.

Prove that the following languages are not regular. You may use the pumping lemma and the closure of the class of regular languages under union, intersection, and complement.

Pumping Lemma:

For a regular language A with the pumping length P, if S is any string of A of length at least P, then the string S can be divided into 3 pieces x, y and z represented as $S = xyz$ should satisfy the following conditions:

1. for each $i \geq 0$, $xy^i z \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

- Consider the Language $L = \{0^n 1^m 0^n | m, n \geq 0\}$.
  Assume that L is regular language and a string $S = 0^P 10^P$.
  Divide the string into three pieces x, y and z. So
  $S = 0^P 10^P = xyz$ where P is the pumping length.
  Obviously that we can always assume $x = 0^{P-K}$,
  $y = 0^(K - K')$ and $z = 0^{K'} 10^P$ $(K > K' \geq 0)$
  Now $xy^0 z = 0^{P-K+K'} 10^P \notin L$ for all possible x,y and z since $K > K'$, which contradicts the pumping lemma and proves that L is not regular.

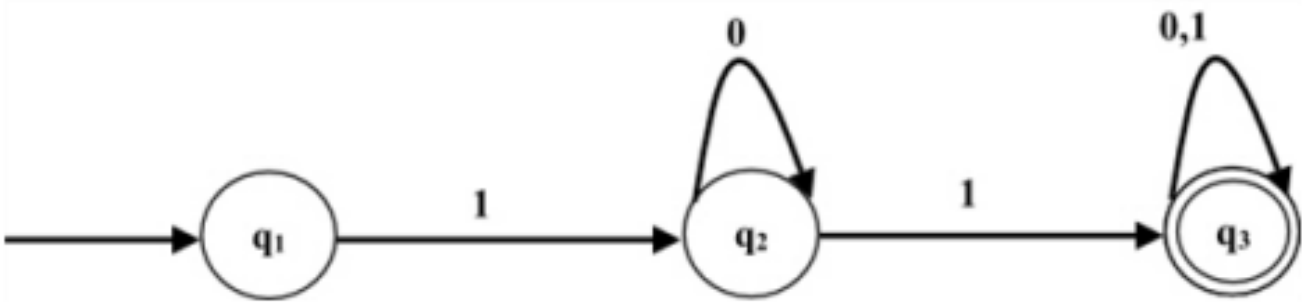
- Consider the Language $L = \{0^m 1^n | m \neq n\}$.
  Assume that L is regular language and a string $S = 0^P 1^{P+P!}$.
  Divide the string into three pieces x, y and z. So
  $S = 0^P 1^P + P! = xyz$ where P is the pumping length.
  Obviously we can always assume that $x = 0^a$, $y = 0^b$ and
  $z = 0^c 1^{P+P!}$, where $a + b + c = P$ and $b \geq 1$.
  Now we take string $s' = xy^{i+1}z$ where $i = \frac{P!}{b}$. Then
  $xyz = 0^{a+b+P!+c} 1^{P+P!} \notin L$, a contradiction since
  $a + b + c + P! = P + P!$. So L is not regular according to
  pumping lemma.

- Consider the Language $L = \{w | w \in \{0,1\}^*$ is not a palindrome$\}$.
  Assume that L is regular language and the compliment of the language L is $\overline{L} = \{w | w \in \{0,1\}^*$ is a palindrome$\}$ is also regular.
  Assume a string $S = 0^P 1 0^P$. Using the construction and induction in problem a, we can obtain a string $xy^0z \notin \overline{L}$. So that the assumption is a contradiction and thus L is not regular by pumping lemma.

- Consider the Language $L = \{wtw | w, t \in \{0,1\}^*\}$.
  Assume that L is regular language.
  Assume a string $S = (01^P)00(01^P)$. Clearly now $w = 01^P$ and $t = 00$. Divide the string into three pieces x,y and z. So $S = xyz = (01^P)00(01^P)$ where P is the pumping length.
  Obviously we always have $z = 1^K0001^P$ and $xy = 01^{P-K}$.
  Clearly the only 0 in xy would be either in x or in y.
  If it is in x, then $xy^0z = 01^{P-K'}0001^P$ with $y = 1^{K'}$. Clearly that $xy^0z \notin L$ because it cannot be written into the form wtw.
  If not, then $x = \epsilon$ and $xy^2z = 01^{P-K}01^P0001^P \notin L$ with the same reason as previous case.
  Thus a contradiction and L is not regular according pumping lemma.

1-49

- Let $B = \{1^k y | y \in \{0,1\}^*$ and y contains at least k 1s, for $k \geq 1\}$.
  Show that B is a regular language.
- It is regular because it can be recognized by the following DFA.



- Let $C = \{1^k y | y \in \{0,1\}^*$ and y contains at most k 1s, for $k \geq 1\}$.
  Show that C is not a regular language.
- Assume that C is a regular language and P the pumping length.
  Consider a string $S = 1^P 01^P \in C$.
  Using pumping lemma, S can be written as $S = xyz$ such that $x = 1^{P-K-K'}$, $y = 1^K$ and $z = 1^{K'}01^P$ with $K > K' \geq 0$.
  Then $xy^0z = 1^{P-K}01^P \in C$ using pumping lemma. Thus $P \leq P - K$ which is not possible since $K > 0$. Thus it is proved that the given language C is not a regular language.

2-4

Give context-free grammars that generate the following languages. In all parts, the alphabet Σ is $\{0, 1\}$.

- $\{w | w$ contains at least three 1s$\}$
  The context-free grammar generating the language is given by
  $S \to P1P1P1P$
  $P \to 0P|1P|\epsilon$
- $\{w | w$ starts and ends with the same symbol$\}$
  The context-free grammar generating the language is given by
  $S \to 0P0|1P1|0|1$
  $P \to 0P|1P|\epsilon$

2-4

- $\{w|w$ the length of w is odd$\}$
  The context-free grammar that generating the language is given by
  $S \to 0|1|00S|01S|10S|11S$
- $\{w|$the length of w is odd and its middle symbol is a $0\}$
  The context-free grammar generating the language is given by
  $S \to 0S0|1S1|0|1S0|0S1$
- $\{w|w$ is a palindrome$\}$
  The context-free grammar generating the language is given by
  $S \to 0S0|1S1|0|1|\epsilon$
- The empty set
  The context-free grammar generating the language is given by
  $S \to S$

2-14

Convert the following CFG into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.

$$A \to BAB|B|\epsilon$$
$$B \to 00|\epsilon$$

Let's add a new start variable S_0 and the rule $S_0 \to A$.
Thus the obtained grammar is

$$\begin{aligned} S_0 &\to A \\ A &\to BAB|B|\epsilon \\ B &\to 00|\epsilon \end{aligned}$$

The addition of new start variable guarantees that the start variable does not occur on the right-hand side of a rule.

Removing all rules that contain ϵ.
Removing $A \to \epsilon$ and $B \to \epsilon$ gives

$$\begin{aligned} S_0 &\to A|\epsilon \\ A &\to BAB|BA|AB|A|B|BB \\ B &\to 00 \end{aligned}$$

The rule $S_0 \to \epsilon$ is accepted since S_0 is the start variable and that is allowed in Chomsky normal form.

2-14

Now remove the unit rules.
Removing $A \to A$ gives

$$\begin{aligned} S_0 &\to A|\epsilon \\ A &\to BAB|BA|AB|B|BB \\ B &\to 00 \end{aligned}$$

Removing $A \to B$ gives

$$\begin{aligned} S_0 &\to A|\epsilon \\ A &\to BAB|BA|AB|00|BB \\ B &\to 00 \end{aligned}$$

Removing $S_0 \to A$ gives

$$\begin{aligned} S_0 &\to BAB|BA|AB|00|BB|\epsilon \\ A &\to BAB|BA|AB|00|BB \\ B &\to 00 \end{aligned}$$

Now replace ill placed terminals 0 by variable U

$$
\begin{aligned}
S_0 &\to BAB|BA|AB|UU|BB|\epsilon \\
A &\to BAB|BA|AB|UU|BB \\
B &\to UU \\
U &\to 0
\end{aligned}
$$

2-14

Now shorten the right-hand side of rules with only 2 variables each. To shorten the rules, replace $S_o \to BAB$ with two rules $S_0 \to BA_1$ and $A_1 \to AB$.
The rule $A \to BAB$ is replaced by $A \to BA_1$ and $A_1 \to AB$.
After replacing these rules, the final context-free grammar in Chomsky normal form is $G = (V, \Sigma, R, S_0)$.
Here the set of variables is $V = \{S_0, A, B, U, A_1\}$, the start variable is S_0. The set of terminals is $\Sigma = \{0\}$, and the rule R is given by

$$
\begin{aligned}
S_0 &\to BA_1|BA|AB|UU|BB|\epsilon \\
A &\to BA_1|BA|AB|UU|BB \\
B &\to UU \\
U &\to 0 \\
A_1 &\to AB
\end{aligned}
$$

This is the final CFG in Chomsky normal form equivalent to the given CFG.

2-20

Let $A/B = \{w | wx \in A \text{ for some } x \in B\}$. Show that if A is context free and B is regular, then A/B is context free.

Hint:

From the problem statement language A is assumed to be context-free and B regular.

From Theorem 2.20: A language is context-free if and only if some push down automaton recognize it.

From Corollary 1.40: A language is regular if and only if some NFA recognize it.

First construct a push down automata P_A for language A:

$$P_A = (Q_A, \Sigma, \Gamma, \delta_A, q_A, F_A)$$

and a NFA M for B:

$$M = (Q_B, \Sigma, \delta_B, q_B, F_B)$$

Assume $P_{A/B}$ is the push down automaton that recognizes A/B:

$$P_{A/B} = (Q_{A/B}, \Sigma, \Gamma^{A/B}, \delta_{A/B}, q_{A/B}, F_{A/B})$$

$P_{A/B}$ is defined as follows:

- $Q_{A/B} = Q_A(Q_A \times Q_B)$
- $\Gamma^{A/B} = \Gamma$
- $q_{A/B} = q_0 P_A$ where $q_0 = q_A = q_B$
- $F_{A/B} = F_A \times F_B$
- For $q_A \in Q_A$

$$\delta_{A/B}(q_A, a, u) = \begin{cases} \delta_A(q_A, a, u), & \text{if } a \in \Sigma \\ \delta_A(q_A, \epsilon, u) \bigcup \{(q_A, q_{B,0}), \epsilon\}, & \text{if } a = \epsilon \end{cases}$$

- For $(q_A, q_B) \in Q_A \times Q_B$

$$\delta_{A/B}((q_A, q_B), a, u) =$$

$$\begin{cases} \emptyset, & \text{if } a \in \Sigma \\ \{((r_A, r_B), v) : (r_A, v) \in \delta_A(q_A, b, u) \text{ and } r_A \in \delta_B(q_B, b)\}, & \text{if } a = \epsilon \end{cases}$$

Therefore it can be claimed that $P_{A/B}$ accepts w iff there occurs a string x such that P_A accepts wx and M accepts x.
For instance, all of w must be read during the first stage for an acceptance calculation of $P_{A/B}$ on input w.
The input symbols that are predicted through the second stage determine a string x recognized by M such that wx is recognized by P_A.
Contrariwise, if w is a string with the property that wx belongs to A for some x belong to B, then there is an acceptance calculation of $P_{A/B}$ where w is read through the first stage, and the input x is predicted in the second stage.
In this instance the P_A components of the states determine an acceptance calculation on P_A on input wx and the M-components of the states determine an acceptance calculation of B on input x.
Hence, it is proved that A/B is CFL by using $P_{A/B}$ from the above discussion.

2-26

Show that if G is a CFG in Chomsky normal form, then for any string $w \in L(G)$ of length $n \geq 1$, exactly $2n - 1$ steps are required for any derivation of w.

It can be proved by applying the Induction method on the string of length n.

- For $n = 1$:
  Consider a string $s = a$ of length 1 in Chomsky normal form, so the valid derivation for this will be $S \rightarrow a$, where $a \in \Sigma$ and S is starting symbol. Clearly it is true that $2n - 1$(for $n = 1$) steps are required to derive a string $s = a$.

- For $n = k$:
  Suppose that for a string of length k in Chomsky normal form, valid derivation will take $2k - 1$ steps.
- For $n = k + 1$, consider a language as follows in CNF where derivation starts with S:

$$S \rightarrow BC$$
$$B \rightarrow *x$$
$$C \rightarrow *y$$

So length of the string starting with S is $|w| = |x| + |y|$ where $|x| > 0$ and $|y| > 0$.
Using the inductive hypothesis, for the above language in CNF of any derivation of string w must be:

$$1 + (2|x| - 1) + (2|y| - 1) = 2(|x| + |y|) - 1$$

Here $n = |x| + |y| = k + 1$.
Hence, the problem statement is proved.

2-30

Use the pumping lemma to show that the following languages are not context-free.

- Consider the language $L = \{0^n1^n0^n1^n | n \geq 0\}$.

Let P be the pumping length of L. To show that L is not a CFL, it is enough to show that string $s = 0^P1^P0^P1^P$ cannot be pumped.
Consider s is of the form $uvxyz$.
If both v and y contain at most one type of alphabet symbol, the string of the form uv^2xy^2z runs of 0s and 1s of unequal length. Hence the string cannot be a member of L.
If either one contains more than one type of alphabet symbol, the string of the form uv^2xy^2z does not contain the symbols in correct order. Hence s cannot be a member of L.
Since s cannot be pumped without violating the pumping lemma condition, L is not a CFL.

- Consider the language $L = \{0^n \# 0^{2n} \# 0^{3n} | n \geq 0\}$.

Let P be the pumping length of L. To show that L is not a CFL, it is enough to show that string $s = 0^P \# 0^{2P} \# 0^{3P}$ cannot be pumped. Consider s is of the form $uvxyz$.
Neither v nor y can contain #, otherwise uv^2xy^2z contains more than 2 #s. Since s is divided into three segments by #s, at least one of the segments 0^P, 0^{2P} and 0^{3P} has no relation with either v or y. Then the length ratio of the segments is not maintained as 1:2:3, uv^2xy^2z is not in L. Hence s cannot be a member of L. Since s cannot be pumped without violating the pumping lemma condition, L is not a CFL.

- Consider the language
  $L = \{w\#t | w \text{ is a substring of } t, \text{ where } w,\ t \in \{a,\ b\}^*\}$.

Let P be the pumping length of L. To show that L is not a CFL, it is enough to show that string $s = a^P b^P \# a^P b^P$ cannot be pumped.
Consider s is of the form $uvxyz$.
Neither v nor y can contain #, otherwise uv^0xy^0z dose not contain #. Hence s cannot be a member of L.
If both v and y are nonempty and occur on the same side of or one of them is empty, there exist uv^kxy^kz such that it is longer on the right-hand side. Hence s cannot be a member of L.
In the remaining case we have v consists of bs and y consists of as. Hence uv^2xy^2z contains more bs on the right-hand side. Hence s cannot be a member of L.
Since s cannot be pumped without violating the pumping lemma condition, L is not a CFL.

- Consider the language $L = \{t_1\#t_2\#...\#t_k | k \geq 2, t_i \in \{a, b\}^*$, and $t_i = t_j$ for some $i \neq j\}$.

Let P be the pumping length of L.
The t_i can be equal to t_j for different i and j values. Hence, the same terms will be appeared in the string s separated by $\#$.
For example, if k value is 2, s can be $ab\#ab$. The language generates the strings that contains same terms comprised of a, b separated by $\#$. The string generated from L are $ab\#ab$, $b\#b\#b$,...etc.
Thus, to show that L is not a CFL, it is enough to redo the proof in question c to show that $s = a^P b^P \# a^P b^P \in L$ cannot be pumped.
Since s cannot be pumped without violating the pumping lemma condition, L is not a CFL.

Say that a language is prefix-closed if all prefixes of every string in the language are also in the language. Let C be an infinite, prefix-closed, context-free language. Show that C contains an infinite regular subset.

Proof.

We apply the principle of pumping lemma for context-free languages on C.
Let the pumping length be P and the string that needs to be considered be s.
We divide the string into $abcde$ such that $ab^k cd^k e \in C$ for all $k \geq 0$ and $|bd| > 0$.
Now it is already given that C is prefix-closed, so $ab^k \in C$ for all $k \geq 0$.
Thus it can be inferred that ab^* is an infinite regular subset of C when $b \neq \epsilon$ and it proves the required statement. □

Let $Y = \{w | w = t_1 \# t_2 \# ... \# t_k \, for \, k \geq 0$, each $t_i \in 1^*$, and $t_i \neq t_j$ whenever $i \neq j\}$. Here $\Sigma = \{1, \#\}$. Prove that Y is not context-free.

Suppose that Y is context-free and a string $s = 1^P\#1^{P+1}\#...\#1^{3P} = uvxyz \in Y$, where P is the pumping length.
Since $|vxy| \leq P$, clearly there would be at most $1\#$ in vxy. Since $uv^kxy^kz \in Y$, so neither v nor y contains $\#$. We assume that $v = 1^a$ and $y = 1^b$. We also assume that there are e 1s at the end of x and f 1s at the beginning of z.
If x contains $\#$, without loss of generality, we suppose $x = 1^c\#1^d$. Clearly $a + b + c + d \leq P - 1$ and $d + f + b = a + c + e + 1 = q$.
Now we consider uv^0xy^0z and uv^2xy^2z. Since $q \in [P, 3P]$, among $q - b$, $q + b$, $q - a - 1$ and $q + a - 1$, at least one of them lies in $[P, 3P]$ and not equals to q or $q - 1$, which conflicts the condition of Y. Thus $\#$ does not exist in x.

Since x does not contain $\#$, vxy is a string consisting of q 1s, where $q \leq P$. If $e + q + f \leq 2P$, then there exist $k \geq 0$ that $e + q + f + ak \in [2P, 3P]$, then $uv^k xy^k z \notin Y$. Similarly, if $e + q + f \geq 2P$, then there exist $k \geq 0$ that $uv^k xy^k z \notin Y$. Thus Y does not satisfy the pumping lemma. Consequently, it is not a CFL.

Let a k-PDA be a pushdown automaton that has k stacks. Thus a 0-PDA is an NFA and a 1-PDA is a conventional PDA. You already know that 1-PDAs are more powerful(recognize a larger class of languages) than 0-PDAs.

3-9

- Show that 2-PDAs are more powerful than 1-PDAs.

Clearly that if a language can be recognized by a 1-PDA, then it can be recognized by a 2-PDA. So 2-PDAs is at least as powerful as 1-PDAs.
Let $L = \{a^n b^n c^n | n \geq 0\}$, where P is the pumping length of L. Now prove that L is not a CFL, it is enough to show that string $s = 1^P 0^P 1^P$ cannot be pumped. So that 1-PDA dose not recognize L.

But the following 2-PDA recognizes L:

1. Push all as that appear in front of the input tape to stack 1;
2. Then push all bs that follows the as in the input stream into stack 2 and if it sees any as in this stage, reject;
3. When it sees the first c, pop stack 1 and stack 2 at the same time. After this, reject the input if it sees any a or b.
4. Pop stack 1 and stack 2 for each input character c it reads. If the input ends when both stacks are empty, accept. Otherwise reject.

Therefore L is recognized by 2-PDA. Hence 2-PDA is more powerful than 1-PDA.

3-9

- Show that 3-PDAs are not more powerful than 2-PDAs.

It is know that 3-PDA is not more powerful than Turing machine, so it is enough to show that 2-PDA can simulate a Turing machine. Now split the tape of TM into two stacks.

1. Stack 1 stores the characters on the left of the head, with the bottom of stack storing the left most character of tape in TM;
2. Stack 2 stores the characters on the right of the head, with the bottom of the stack storing the right most character of tape in TM.

Show the Transition example for left and right.

Left Transition:

1. If $\delta(q_i, w_i) = (q_j, w_j, L)$, then w_i on top of second stack.
2. If \$ is on top of stack then reject and move the left edge of tape. Otherwise pop w_i from top of second stack. Push w_j in second stack.
3. Pop the symbol of the first stack and push it to second stack. Switching the state of q_j.
4. If q_j is accept state, then accepts w. Otherwise rejects.

Right Transition:

1. If $\delta(q_i, w_i) = (q, w_j, R)$, then w_i on top of second stack.
2. Pop w_i from second stack and push w_j in second stack.
3. If \$ is on top of second stack, push null on to second stack. Switch the state of q_j.
4. If q_j is accept state, then accepts w. Otherwise rejects.

In this way 2-PDA simulates Turing machine. Hence 3-PDA is not more powerful than 2-PDA.

3-14

A queue automaton is like a push-down automaton except that the stack is replaced by a queue. A queue is a tape allowing symbols to be written only on the left-hand end and read only at the right-hand end. Each write operation (we'll call it a push) adds a symbol to the left-hand end of the queue and each read operation (we'll call it a pull) reads and removes a symbol at the right-hand end. As with a PDA, the input is placed on a separate read-only input tape, and the head on the input tape can move only from left to right. The input tape contains a cell with a blank symbol following the input, so that the end of the input can be detected. A queue automaton accepts its input by entering a special accept state at any time. Show that a language can be recognized by a deterministic queue automaton iff the language is Turing-recognizable

The equivalence between the queue automaton Q and the Turing machine M is required to be shown.

The automaton can be simulated by the Turing machine as follows:

Consider the entire tape as a queue. One by one each symbol is altered and the movement of the tape takes place to the right. If more than one numbers are to be pushed in the queue, then it is done by shifting contents to the right. If the end of the tape if reached, the leftmost symbol of the tape is approached.

Reversely, the Turing machine can be simulated by the automaton as follows:

The alphabet of the Turing machine M is expanded by inserting an extra symbol. A left end marker # is inserted to the queue. The symbols are pushed to left and read (popped) from the right.

3-19

Show that every infinite Turing-recognizable language has an infinite decidable subset.

Let L be an infinite Turing recognizable language. We know that a language is Turing recognizable iff some enumerator enumerates it. So let M be the enumerator that enumerates L.
Consider a language $L' = \{w_1, w_2, ..\}$ where

1. w_1 is the first string enumerated by M
2. for every $i > 1$, w_i is the first string enumerated by M and that is lexicographically larger than w_{i-1}

- First we prove L' is infinite and is subset of L:

Assume that L' is finite, then w_i, then it exist a k such that w_k is the last and lexicographically largest element in L' and all strings enumerated by M must be lexicographically less than w_k. Since there are only a finite number of strings that are less than w_k, L would then be finite. This is a contradiction. Hence L' is infinite. All strings in L' were at some point enumerated by M, so clearly L' is subset of L.

- Next we have to prove that L' is decidable:

Now we will show that L' is decidable for the given enumerator M, we can construct an enumerator M' in lexicographic order which does the following:

1. Let w be the last string that M' emitted, and initialize w to a dummy value that comes lexicographically before all strings.
2. Simulate M until it emits a string t.
3. If $t > w$ lexicographically, then set $w = t$ and let M' emit t, else ignore t.
4. Resume simulating M, and go to step 2

Thus M' is constructed and that enumerated L' in lexicographic order. So that L' is decidable.
So finally L' is an infinite decidable subset of L.

4-2

Consider the problem of determining whether a DFA and a regular expression are equivalent. Express this problem as a language and show that it is decidable.

4-2

Express the language as $\{L = \langle R, S \rangle \,|R$ is a DFA and S is a regular expression with $L(R) = L(S)\}$.
Recollect the Theorem 4.5: $EQ_{DFA} = \{\langle A, B \langle \,|A$ and B are DFA and $L(A) = L(B)\}$

4-2

Assume that M is the Turing machine that decides L. M can be defined as follows:
For input $L = \langle R, S \rangle$, where R is DFA and S is a regular expression:
Convert S into a DFA D_S using the algorithm in the proof of Kleene's Theorem.
Operate a Turing machine as a decider F using Theorem 4.5 on input $\langle R, D_S \rangle$.
If F accepts, accept L. Else reject L.

Let $A_{\epsilon CFG} = \{\langle G \langle | G \text{ is a CFG that generates } \epsilon\}$. Show that $A_{\epsilon CFG}$ is decidable

- For showing that $A_{\epsilon CFG}$ is decidable, build a Turing machine M for deciding it. For all CFG G:
- if G derives ϵ then $M(\langle G \langle)$ accepts;
- else $M(\langle G \langle)$ rejects.

Now we construct it.
First convert G into an equivalent G' in CNF. If $S \to \epsilon$ is in G' then G' derives ϵ. So do G since $L(G) = L(G')$.
As G' is in CNF, the only possible ϵ-rule in G' is $S \to \epsilon$. Thus:
Turing machine $T =$ on input $\langle G \langle$ where G is a CFG.

1. Convert G to G'.
2. If G' contains $S \to \epsilon$ then accept it.
3. Otherwise reject it.

Such construction shows that $A_{\epsilon CFG}$ is decidable.

4-11

Let $INFINITE_{PDA} = \{\langle M \langle \mid M \text{ is a PDA and } L(M) \text{ is an infinite language}\}$. Show that it is decidable

The decidability of $INFINITE_{PDA}$ is as follows:
Convert the given PDA M into a CFG G, then convert it into an equivalent grammar G' in CNF.
Generate a graph A_1: each variable is a vertex, and for a rule or production $T \to UV$, add edges (T, U) from T to U, and (T, V) from T to V.
Apply DFS or BFS from the start state S to see if the graph has a directed cycle. If does, accept it. Otherwise reject.
Now prove $L(G')$ is infinite. Following the proof of pumping lemma, V can derive sVt for some string s, t, which implies that A_1 has a cycle involving V. Since there is a scope for finding a cycle from S, there is a rule $S \to {}^*aVb$. Thus, $S \to {}^*au^iVv^ib$ for all $i \geq 0$ and so $L(M)$ is infinite.
Hence $INFINITE_{PDA}$ is decidable.