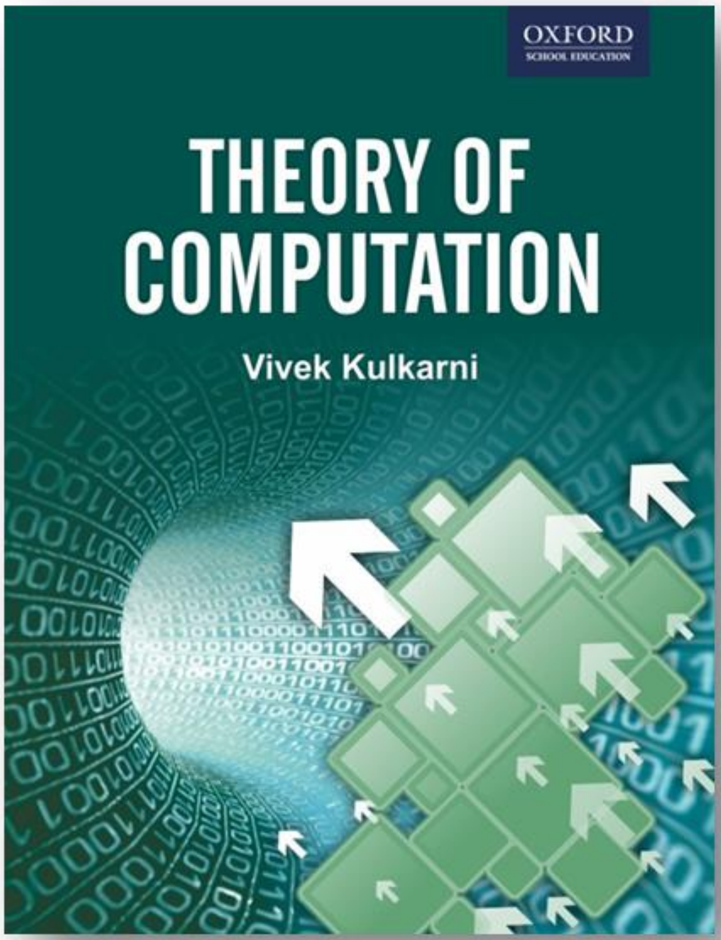


Author: Vivek Kulkarni
(vivek_kulkarni@yahoo.com)

Solution for

Model Question Paper 5

(From Appendix B)

SECTION I

Q.1 (a) (i) Basic machine: Refer to the section 2.1.1.

(ii) Moore machine: Refer to the section 2.10.1.

(iii) Mealy machine: Refer to the section 2.10.2.

Q.1 (b) (i) NFA without $\in$-moves: Refer to the example 2.11 from the book.

(ii) DFA: Refer to the example 2.12 from the book.

OR

Q.1 (a) Refer to the example 2.20 from the book.

Q.1 (b) Refer to the example 2.25 from the book.

Q.2 (a)

(i) If ($L1 \subseteq L2$) and $L1$ is not regular, then $L2$ is not regular.

The statement is not always true, that means, it is false. Let us consider the example of following languages $L1$ and $L2$,

Let, $L1$ = $\{0^n1^n \mid n >= 0\}$

= $\{\in, 01, 0011, 000111, \ldots.\}$

$L1$ here is not a regular language; $L1$ actually is a CFL.

Let, $L2$ = 0^*1^*

= $\{\in, 0, 1, 00, 01, 10, 11, 000, \ldots, 0011, \ldots\}$

$L2$ is a regular language as we know.

Thus, even though ($L1 \subseteq L2$) and $L1$ is not regular, $L2$ is regular.

Hence, the statement is false.

(ii) If ($L1 \subseteq L2$) and $L2$ is not regular, then $L1$ is not regular.

The statement is false.

Let us consider the example of following languages $L1$ and $L2$,

Let, $L2$ = {set of all palindrome strings over {0, 1}}

= $\{\in, 0, 1, 00, 11, 000, 010, 101, 111, 0000, \ldots\}$

$L2$ here is not a regular language; $L2$ actually is a CFL.

Let, $L1 = \{\in, 0, 1, 00, 11, 000, 111, 0000, \ldots\}$

$L1$ thus contains strings consisting of all 0's or 1's or an empty string. $L1$ is actually a regular language and we can denote it using a regular expression, $r = 0^* + 1^*$.

Thus, even though ($L1 \subseteq L2$) and $L2$ is not regular, $L1$ is regular.

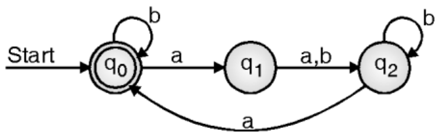
Hence, the statement is false.

(iii) If $L1$ and $L2$ are not regular, then ($L1 \cup L2$) is not regular.

The statement is true. As we know most of the languages are closed under union. For example, if we take union of two CFLs the result is also a CFL.

Q.2 (b) Refer to the example 3.45 from the book.

Q.2 (c) We are asked to construct a regular expression for the following DFA:



The state equations pertaining to the three states, q_0, q_1, and q_2, are:

$q_0 = q_0 \text{ b} + q_2 \text{ a} + \in$ (q_0 is initial as well as final state; hence $\in$ is added)

$q_1 = q_0 \text{ a}$

$q_2 = q_2 \text{ b} + q_1 (\text{a} + \text{b})$

As per Arden's theorem,

If $R = RQ + P$ then, R can be written as, $R = PQ^*$

If we apply Arden's theorem to q_2 with $R = q_2$, $Q = \text{b}$ and $P = q_1 (\text{a} + \text{b})$ we get,

$$q_2 = q_1 (\text{a} + \text{b}) \text{ b}^*$$

If we substitute for q_1 in q_2 we get,

$$q_2 = q_0 \text{ a} (\text{a} + \text{b}) \text{ b}^*$$

As q_0 is the final state we need to solve for q_0 which will give us the regular expression for the DFA given. Let us substitute the value of q_2 in q_0, we get,

$$q_0 = q_0 \text{ b} + q_2 \text{ a} + \in$$

$$= q_0 \text{ b} + q_0 \text{ a} (\text{a} + \text{b}) \text{ b}^* \text{a} + \in$$

$= q_0 \, (b + a\,(a + b)\, b^*a) + \in$

Therefore, by applying Arden's theorem,

$q_0 = \in (b + a\,(a + b)\, b^*a)^*$

$= (b + a\,(a + b)\, b^*a)^*$

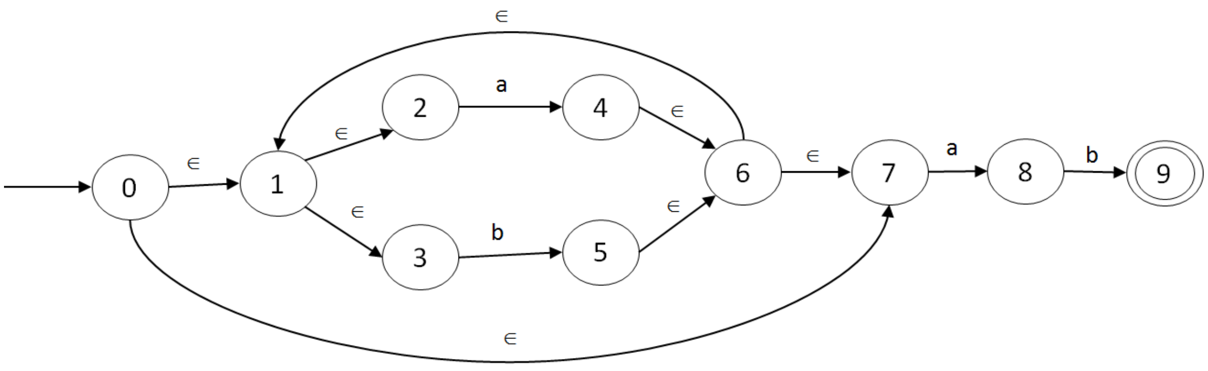
OR

Q.2 **(a)** It is expected to construct a DFA that recognizes the regular set:

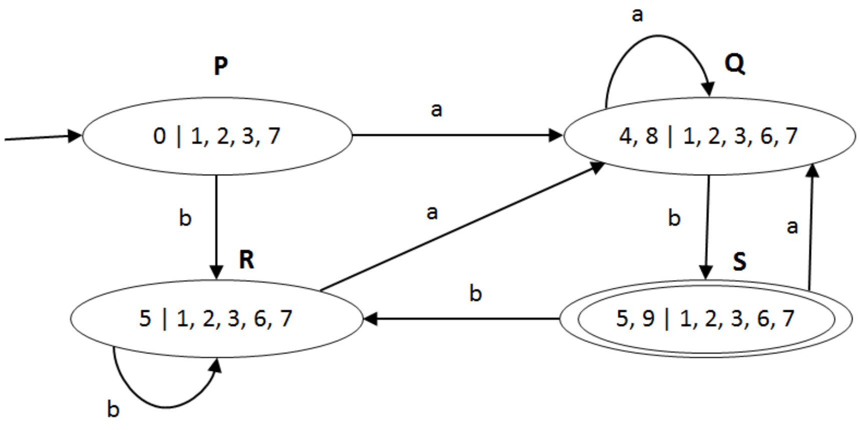
$R = (a/b)^* \cdot a \cdot b$

Let us first build the NFA with ∈-moves and the convert the same to DFA.

The TG for NFA with ∈-moves is as follows,



Let us convert this NFA with ∈-moves to its equivalent DFA using a direct method.



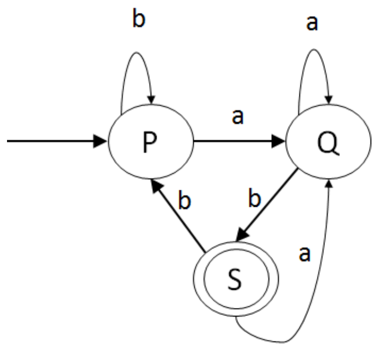
We have relabelled the states as well. Let us see if we can minimize it. The STF for the DFA looks like,

Q\∑	a	b
P	Q	R
Q	Q	S
R	Q	R
* S	Q	R

We can see that states *P* and *R* are equivalent. Hence, we can replace *R* by *P* and get rid of *R*. The reduced STF is,

Q\∑	a	b
P	Q	P
Q	Q	S
* S	Q	P

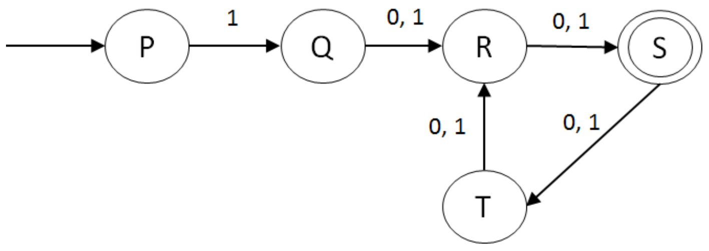
The TG for the equivalent DFA is,



Q.2 **(b)** The given language is,

$L = \{x \in \{0, 1\}^* \mid x$ starts with 1; and $\mid x \mid$ is divisible by 3$\}$

The DFA for the same looks like,



One can see the on reading symbol 1 the DFA makes a move to state *Q*. This makes sure that every string starts with 1 always. The path *PQRS* reads the string of length 3. Later *STRS* keeps on reading string of length 3 always. This makes sure that the length of any input accepted by the DFA is always divisible by 3. The regular expression for the language can be written as,

$$1\ (0 + 1)\ (0 + 1)\ [(0 + 1)\ (0 + 1)\ (0 + 1)]^*$$

Q.2 **(c)** Refer to the section 2.14.

Q.3 **(a)** Given grammar, $G = \{(S, A), (a, b), P, S\}$, where P consists of

$$S \rightarrow \underset{1}{a\,A\,S} \mid \underset{2}{a}$$

$$A \rightarrow \underset{3}{S\,b\,A} \mid \underset{4}{S\,S} \mid \underset{5}{b\,a}$$

Let us derive the string '*aabbaa*' using leftmost derivation as we as rightmost derivation.

Leftmost derivation:

$$\begin{aligned}
S &\Rightarrow a\,\underline{A}\,S && \text{, rule (1)}\\
&\Rightarrow a\,\underline{S}\,b\,A\,S && \text{, rule (3)}\\
&\Rightarrow a\,a\,b\,\underline{A}\,S && \text{, rule (2)}\\
&\Rightarrow a\,a\,b\,b\,a\,\underline{S} && \text{, rule (5)}\\
&\Rightarrow a\,a\,b\,b\,a\,a && \text{, rule (2)}
\end{aligned}$$

Rightmost derivation:

$$\begin{aligned}
S &\Rightarrow a\,A\,\underline{S} && \text{, rule (1)}\\
&\Rightarrow a\,\underline{A}\,a && \text{, rule (2)}\\
&\Rightarrow a\,S\,b\,\underline{A}\,a && \text{, rule (3)}\\
&\Rightarrow a\,\underline{S}\,b\,b\,a\,a && \text{, rule (5)}\\
&\Rightarrow a\,a\,b\,b\,a\,a && \text{, rule (2)}
\end{aligned}$$

Q.3 **(b)** Following is the grammar given to be converted to CNF,

$S \rightarrow a\,S\,a \mid b\,S\,b \mid a \mid b \mid a\,a \mid b\,b$

In CNF you can either have 2 non-terminals or a single terminal on the right hand side of every production.

Replacing a by A wherever appropriate we get,

$S \rightarrow \mathrm{A}\,S\,A \mid B\,S\,B \mid a \mid b \mid A\,A \mid \mathrm{B\,B}$

$\mathrm{A} \rightarrow \mathrm{a}$

$\mathrm{B} \rightarrow \mathrm{b}$

Let, $C \rightarrow AS$ and $D \rightarrow BS$

We get the CFG expressed in CNF as below,

$S \rightarrow C\,A \mid D\,B \mid a \mid b \mid A\,A \mid B\,B$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow AS$

$D \rightarrow BS$

Q.3 **(c)** Given left-linear grammar is,

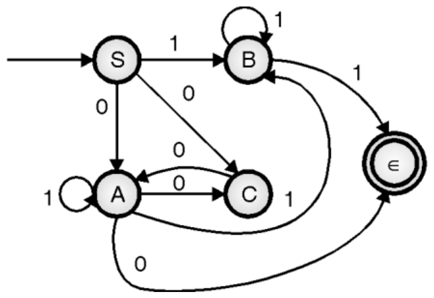
$S \rightarrow B\,1 \mid A\,0 \mid C\,0$

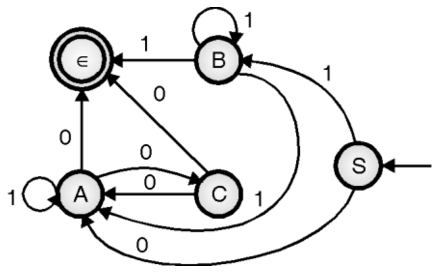
$A \rightarrow C\,0 \mid A\,1 \mid B\,1 \mid 0$

$B \rightarrow B\,1 \mid 1$

$C \rightarrow A\,0$

Let us obtain the equivalent right-liner grammar, G_R. We need to first convert the FA corresponding to the given grammar; then we need to reverse the positions of the start and the final state and reverse all the transitions.





Using the TG after reversal, we can write the equivalent G_R as:

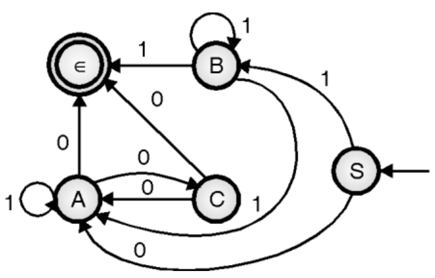
$S \rightarrow 0\,A \mid 1\,B$

$A \rightarrow 1\,A \mid 0\,C \mid 0$

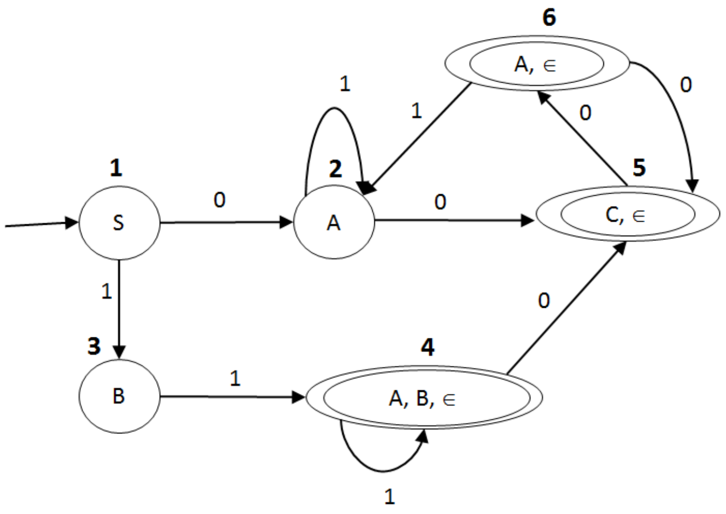
$B \rightarrow 1\,B \mid 1\,A \mid 1$

$C \rightarrow 0 \mid 0\,A$

Let us now obtain the DFA using the NFA we obtained above after reversal.



Let us use the direct method of conversion to obtain the equivalent DFA.



We have relabelled the states. As one can see this DFA cannot be reduced further and hence, is the final answer.

OR

Q.3 **(a)** Refer to the example 5.29 from the book.

Q.3 **(b)** Refer to the section 5.17.

Q.3 **(c)** (i) Backus–Naur form: Refer to the section 5.19.

(ii) Simplification of CFG: Refer to the section 5.9.

(iii) Leftmost and rightmost derivations: Refer to the section 5.5.

(iv) Context sensitive grammar: Refer to the section 5.11.2

SECTION II

Q.4 **(a)** (i) Acceptance of a CFL by empty stack by a PDA: Refer to the section 6.5.2.

(ii) Acceptance of a CFL by final state by a PDA: Refer to the section 6.5.3.

Q.4 **(b)** Refer to the example 6.12 from the book.

Q.4 **(c)** Refer to the section 6.3.

OR

Q.4 **(a)** Refer to the example 6.3 from the book.

Q.4 **(b)** The CFG to be converted to PDA is,

$S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid bS \mid aBB$

Let us rewrite the CFG in the CNF as,

$S \to XB \mid YA$

$A \to a \mid XS \mid YR$

$B \to b \mid YS \mid XP$

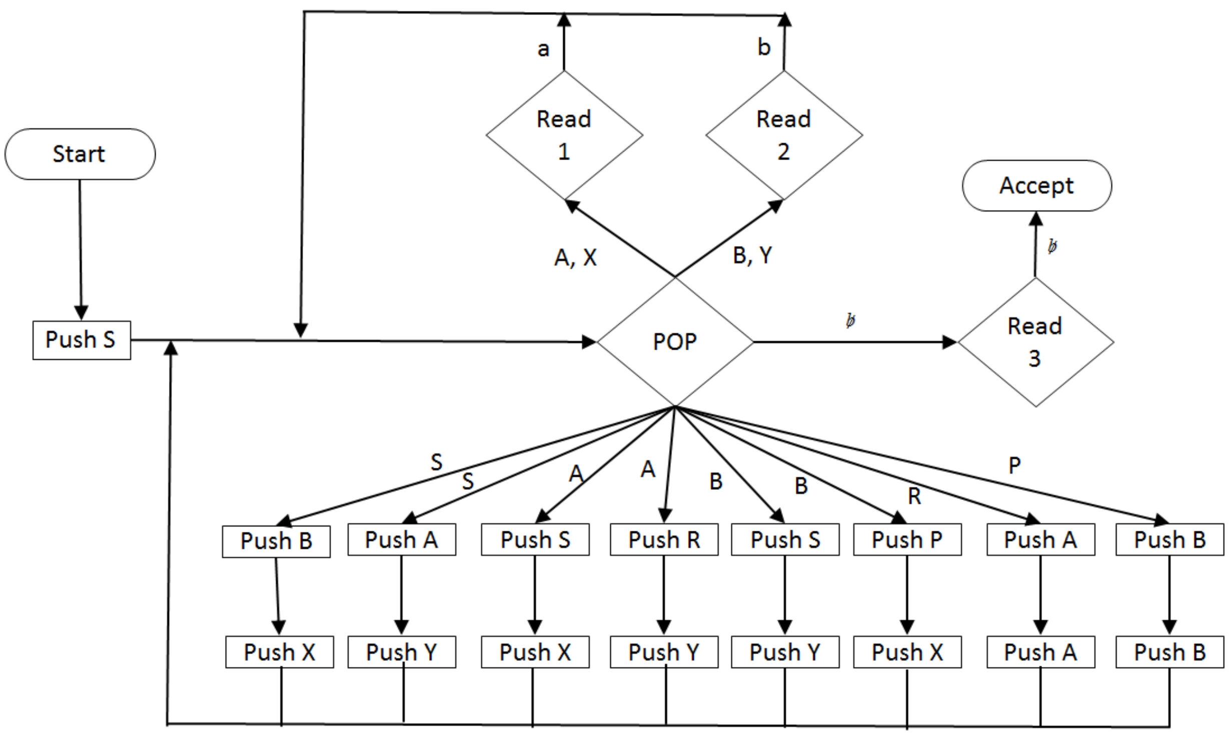
$R \to AA$

$P \to \mathrm{BB}$

$X \to a$

$Y \to b$

The PDA equivalent to the CFG is,



Q.5 **(a)** Refer to the example 4.11 from the book.

Q.5 **(b)** (i) Solvability: Refer to the section 4.13.
(ii) Semi-solvability: Refer to the section 4.13.
(iii) Unsolvability: Refer to the section 4.13.

Q.5 **(c)** Refer to the example 8.6 from the book.

OR

Q.5 **(a)** Refer to the example 4.7 from the book.

Q.5 **(b)** Refer to the example 8.4 from the book..

Q.5 **(c)** Refer to the section 4.12.

Q.6 **(a)** Refer to the example 7.2 from the book for construction of canonical-LR parser. We can reduce the parsing table of example 7.2 to obtain the reduced LALR parsing table as shown below. The state pairs that have the same transitions but different look-ahead are – (4, 11), (5, 12), (7, 13), and (8, 14). The LALR parsing table is,

State	***ACTION***				***GOTO***		
	id	=	*	$	*S*	*L*	*R*
0	*s*5	–	*s*4	–	1	2	3
1	–	–	–	accept	–	–	–
2	–	*s*6	–	*r*5	–	–	–
3	–	–	–	*r*2	–	–	–
4,11	*s*5,12	–	*s*4,11	–	–	8,14	7,13
5,12	–	*r*4	–	*r*4	–	–	–
6	*s*12	–	*s*11	–	–	10	9
7,13	–	*r*3	–	*r*3	–	–	–
8,14	–	*r*5	–	*r*5	–	–	–
9	–	–	–	*r*1	–	–	–
10	–	–	–	*r*5	–	–	–

Q.6 **(b)** Refer to the example 11.19 from the book.

OR

Q.6. **(a)** Given CFG is,

$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid E \wedge E \mid (E) \mid id$

As the grammar is ambiguous, let us first remove the ambiguity. The equivalent unambiguous grammar is,

$E \rightarrow E + T \mid \mathrm{E} - \mathrm{T} \mid T$

$T \rightarrow T * F \mid \mathrm{T} / \mathrm{F} \mid F$

$F \rightarrow G \wedge F \mid G$ (^ is a right associative operator)

$\mathrm{G} \rightarrow (\mathrm{G}) \mid id$

We need to remove the left recursion as well. Removing the left recursion we get,

$E \rightarrow T E'$

$E' \rightarrow + T E' \mid - T E' \mid \epsilon$

$T \rightarrow F T'$

$T' \rightarrow * F T' \mid / F T' \mid \epsilon$

$F \rightarrow G \wedge F \mid G$

$\mathrm{G} \rightarrow (\mathrm{G}) \mid id$

We need to do some left factoring for the productions, $F \rightarrow G \wedge F \mid G$ as,

$F \rightarrow GH$

$H \rightarrow \wedge F \mid \in$

Thus, the final simplified grammar required for top-down parsing is,

$E \rightarrow T E'$

$E' \rightarrow + T E' \mid - T E' \mid \in$

$T \rightarrow F T'$

$T' \rightarrow * F T' \mid / F T' \mid \in$

$F \rightarrow GH$

$H \rightarrow \wedge F \mid \in$

$G \rightarrow (G) \mid id$

Q.6. (b) Refer to the section 10.2.1.

Q.6. (c) Refer to the section 9.9.

Q.6. (d) Refer to the section 9.12.