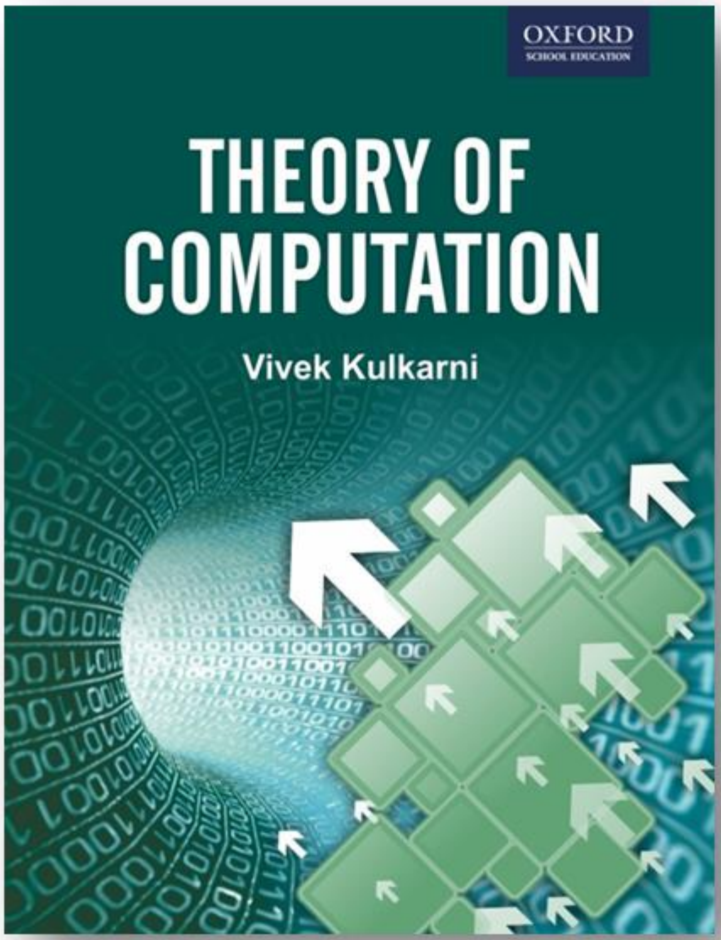


Author: Vivek Kulkarni
(vivek_kulkarni@yahoo.com)

Solution for

Model Question Paper 2

(From Appendix B)

Q.1 a) (i) Alphabet: Refer to the section 1.2.2.

(ii) Symbol: Refer to the section 1.2.1.

(iii) Language: Refer to the section 1.6.

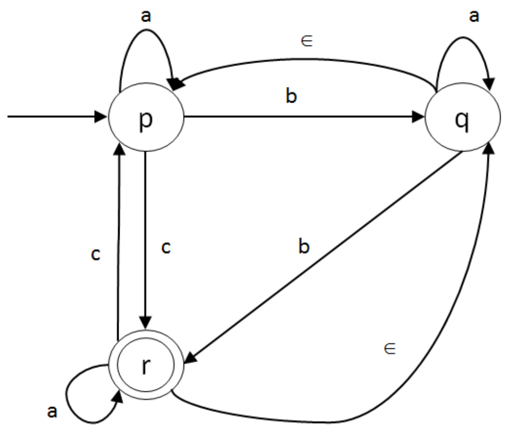
(iv) Word: Refer to the section 1.2.3.

(v) Closure of a set: Refer to the section 1.3.1.9.

(vi) Reflexive and transitive closure of a relation: Refer to the section 1.4.2.2.

Q.1 b) Refer to the example 2.21 from the book.

Q.1 c) Let us first draw the TG for the given NFA with ∈-transitions.

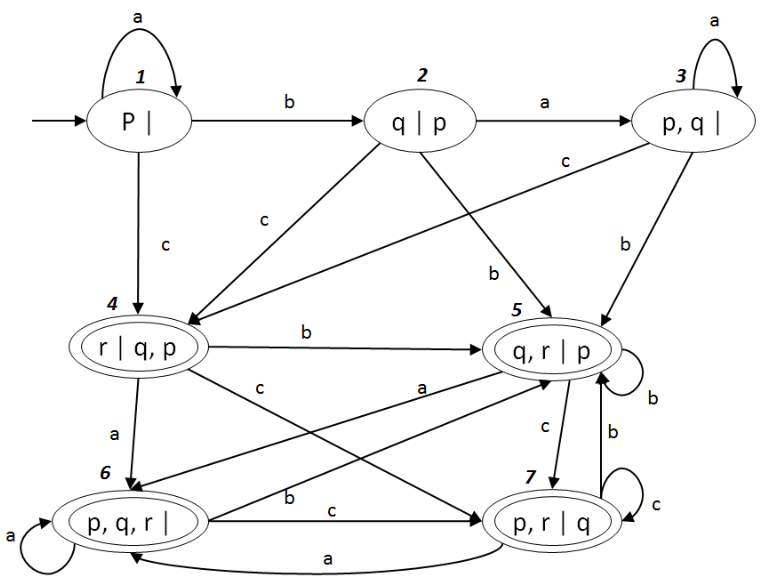


1) ∈-closure (p) = {p}
   ∈-closure (q) = {p, q}
   ∈-closure (r) = {p, q, r}

2) All the strings of length three or less accepted by the automata
   = {c, ac, bb, bc, ca, cb, cc, abb, abc, bab, bac, aca, acb, acc, caa, cbb, ccc, cab, cba, cac, cca, cbc, ccb}

3) The equivalent DFA can be obtained using a direct method as below. Each state has the primary as well as the secondary label. Secondary label consists of all the reachable states from the given state. We also have relabelled the states from 1 to 7, out of which 4, 5, 6 and 7 are final states.

   Let us also refer to the STF drawn and see whether it can be reduced.

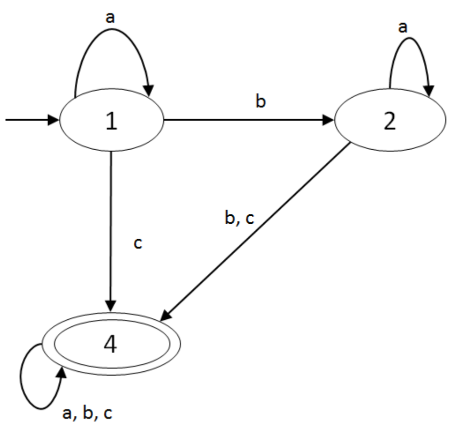


	a	b	c
1	1	2	4
2	3	5	4
3	3	5	4
* 4	6	5	7
* 5	6	5	7
* 6	6	5	7
* 7	6	5	7

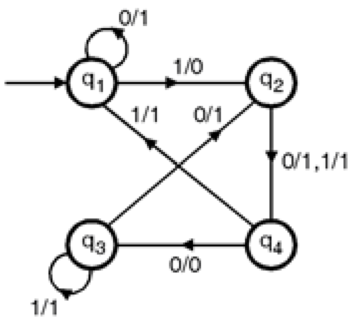
One can see that states 4, 5, 6 and 7 are equivalent. Hence, we can keep only 4 and remove the others by replacing all their occurrence by 4. Similarly, 2 and 3 are also equivalent; hence, 3 can be removed in lieu of state 2. Therefore, the reduced STF is,

	a	b	c
1	1	2	4
2	2	3	4
* 4	4	4	4

The reduced state DFA is,



Q.1 d) Given mealy machine is,



Q/Σ	0	1
q_1	q_1	q_2
q_2	q_4	q_4
q_3	q_2	q_3
q_4	q_3	q_1

STF

Q/Σ	0	1
q_1	1	0
q_2	1	1
q_3	1	1
q_4	0	1

MAF

The STF for the equivalent Moore machine can be written as,

Q/Σ	0	1
q_1,0	q_1,1	q_2,0
q_1,1	q_1,1	q_2,0
q_2,0	q_4,1	q_4,1
q_2,1	q_4,1	q_4,1
q_3,0	q_2,1	q_3,1
q_3,1	q_2,1	q_3,1
q_4,0	q_3,0	q_1,1
q_4,1	q_3,0	q_1,1

STF

Q/Σ	Output
q_1,0	0
q_1,1	1
q_2,0	0
q_2,1	1
q_3,0	0
q_3,1	1
q_4,0	0
q_4,1	1

MAF

Q.2 a)

i) Given $n \geq 0$

For n = 1,	$0^2 = 00$,	length = 2
For n = 2,	$0^4 = 0000$,	length = 4
For n = 3,	$0^6 = 000000$,	length = 6

Now, from observation we can find out the property of the language given and is that it consists of strings having even length and all 0's. It is a regular language denoted by a regular expression $(0\ 0)^*$.

ii) Given language definition is,

$\{0^m\ 1^n\ 0^{m+n} \mid m \geq 1 \text{ and } n \geq 1\}$

Here, we can see that number of 0's at the end is the sum of 'm' number of 0's and 'n' number of 1's read already. This requires remembering 'm' and 'n' which requires memory. Finite state machines which accepts regular languages does not have any memory. Hence, the language is not regular.

Q.2 **b)** We need to show that the following language is non-regular using Pumping lemma,

$L = \{0^p\ 1^p\ p^{p+q} \mid p \geq 1, q \geq 1\}$

As we observe the length of every string from the language L = 2p + 2q = 2 (p + q) is even.

Step 1: Let us assume that the language L is a regular language. Let n be the constant of pumping lemma.

Step 2: Let us choose a sufficiently large string z, such that $z = xx$, where $x = 0^p1^q\mathrm{P}^{p+q}$, for some large $p, q > 0$; the length of z is given by: $|z| = 2\ (p + q) \geq n$.

Since we assumed that L is a regular language and from the language definition it is an infinite language, we can now apply pumping lemma. Hence, we should be able to write z as: $z = uvw$.

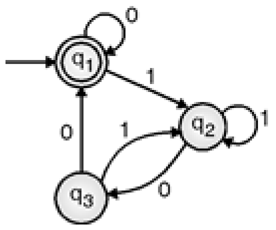
Step 3: As per pumping lemma, every string 'uv^iw', for all $i \geq 0$, is in L. Further, $|v| \geq 1$, which means that v cannot be empty, and must contain one or more symbols.

Let us consider the case when v contains a single symbol from {0, 1}. We assume $z = uvw = xx = 0^p1^q\mathrm{P}^{p+q}0^p1^q\mathrm{P}^{p+q}$. As per pumping lemma, we would expect 'uv^2w' also to be a member of L. However, this cannot be the case as v contains only a single symbol; hence, pumping v would cause the first x in string 'xx' to end with v, and the second x of string 'xx' to begin with v. For example, for $z = 0^p1^q\mathrm{P}^{p+q}0^p1^q\mathrm{P}^{p+q}$, after pumping $v = 0$ once, we get, $z_1 = 0^p1^q\mathrm{P}^{p+q}\ 00\ 0^p1^q\mathrm{P}^{p+q}$, which cannot be represented as a concatenation of two equal sub-strings. Thus, uv^2w is not a member of L, as it modifies the string of the form xx to $xvvx$ rather than $xvxv$. This contradicts our assumption that L is regular.

Let us now consider the case when *v* contains both the symbols, i.e., 0 as well as 1. The sample *v* could be written as 01, or 100, and so on. When we try to pump *v* multiple times, we obtain strings of the form, xv^2v^2x, xv^3v^3x, and so on, which is against the language definition *xx*—every string is represented as concatenation of two equal sub-strings. Thus, uv^iw, for all $i \geq 0$ is not a member of *L*. This contradicts our assumption that *L* is regular.

Hence, language *L* is non-regular.

Q.2 **c)** Given FA is,



There is only one initial state which is also a final state. Hence,

$q_1 = q_1 0 + q_3 0 + \in$

$q_2 = q_1 1 + q_2 1 + q_3 1$

$q_3 = q_2 0$

Substituting for q_3 in q_2 we get,

$q_2 = q_1 1 + q_2 1 + (q_2 0)1$

$= q_1 1 + q_2 (1 + 01)$

By applying the Arden's theorem we get,

$q_2 = q_1 1 (1 + 01)^*$

Substituting for q_3 in q_1 we get,

$q_1 = q_1 0 + q_3 0 + \in$

$= q_1 0 + q_2 00 + \in$

$= q_1 (0 + 1(1+01)^* 00) + \in$

Once again applying theorem,

$q1 = \in (0 + 1(1 + 01)^* 00)^*$

$= (0 + 1(1 + 01)^* 00)^*$

As q_1 is the only final state, the regular expression corresponding to the given FA is,

$(0 + 1(1 + 01)^* 00)^*$

Q.3 **a)** Refer to the section 5.10.2.

Q.3 **b)** Refer to the section 5.8.

Q.3 **c)** Given grammar is,

$S \rightarrow aA \mid bB \mid aS \mid a$

$A \rightarrow bA \mid \in$

$B \rightarrow aB \mid \in$

As there is a self-loop on 'S', let us rewrite the grammar as below,

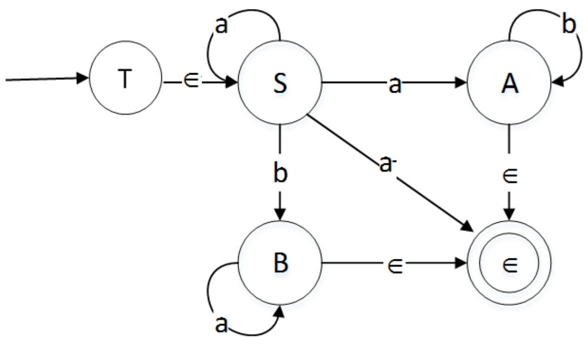
$T \rightarrow S$

$S \rightarrow aA \mid bB \mid aS \mid a$

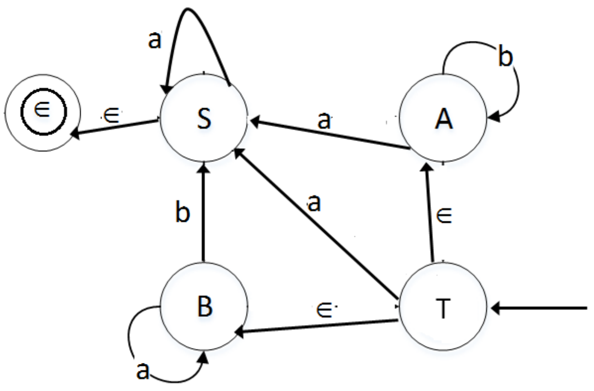
$A \rightarrow bA \mid \in$

$B \rightarrow aB \mid \in$

TG representing G_R



TG after reversing the transitions and interchanging initial and final vertices.



The equivalent G_L can be written as,

$T \rightarrow A \mid B \mid Sa$

$A \rightarrow Ab \mid Sa$

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

$S \rightarrow S\,a \mid \in$

Q.3 d) The CFG can be written as,

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

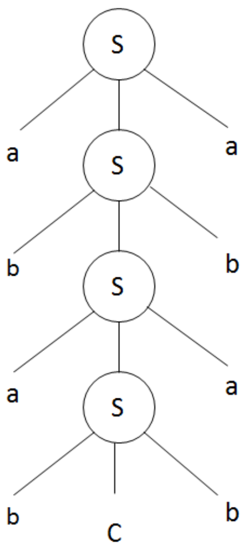
This is nothing but a CFG for all the palindrome strings with middle symbol as 'C'.

Leftmost derivation for '*ababCbaba*':

$$\begin{aligned} S &\Rightarrow a\,\underline{S}\,a \\ &\Rightarrow a\,b\,\underline{S}\,b\,a \\ &\Rightarrow a\,b\,a\,\underline{S}\,a\,b\,a \\ &\Rightarrow a\,b\,a\,b\,\underline{S}\,b\,a\,b\,a \\ &\Rightarrow a\,b\,a\,b\,C\,b\,a\,b\,a \end{aligned}$$

As S is the only non-terminal on the right hand side of any production, even the rightmost derivation looks exactly same.

The derivation tree is as shown below,



Q.3 e) $L = \{a^n\,b^m \mid n \neq m\}$ is a CFL. The CFG for the same can be written as,

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

$A \rightarrow a\,A \mid \in$

$B \rightarrow b\,B \mid \in$

As $n \neq m$, n can either be greater than m or less than m.

'$S \rightarrow a\,S\,b$' always generates equal number of *a*'s and *b*'s. Once that is done, the middle S can either be derived as '$S \rightarrow a\,A$' *or* '$S \rightarrow b\,B$' which generates more number of *a*'s than *b*'s which is, $n > m$ or more number of *b*'s than *a*'s which is, $n < m$.

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

Q.4 b) Refer to the section 6.8.

Q.4 c) Let us write CFG for the language given,

$L = \{\omega_1 c \omega_2 : \omega_1, \omega_2 \in \{a, b\}^*; \omega_1 \neq \omega_2\}$

As 'ω_1' and 'ω_2' cannot be same, we can consider the scenario when 'ω_1' starts with a, 'ω_2' starts with b and vice versa. Same can be the case that they end with different symbols. We can write CFG with the above considerations as below,

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

$A \rightarrow a\,A \mid b\,A \mid \in$

We can obtain the NPDA from the above CFG as,
Use algorithm discussed in the section 6.7 to convert the above CFG to NPDA.

Q.5 a)

(i) The transition table for the required TM can be drawn as follows:

Input / State	a	b	X	Y	Δ
q_0	(q_1, X, R)	(HALT, b, N)	-	-	–
q_1	(q_1, a, R)	(q_2, b, R)	-	-	–
q_2	-	(q_2, b, R)	-	(q_2, Y, R)	(q_3, Δ, L)
q_3	-	(q_4, Y, L)	-	(q_3, Y, L)	-
q_4	(q_5, a, L)	(q_4, b, L)	-	-	-
q_5	(q_5, a, L)	-	(q_0, X, R)	-	-

(ii) Refer to the example 4.2 from the book.

Q.5 b) Refer to the sections 4.1, 6.1, 8.1, 8.3 and 8.5.

Q.6 a) (i) Multi-track Turing machine: Refer to the section 4.12.

(ii) Multi-tape Turing machine: Refer to the section 4.10.

(iii) Recursively enumerable language: Refer to the section 4.15.

(iv) Recursive language: Refer to the section 4.15.

Q.6 b) Refer to the sections 4.14 for halting problem description and for showing that it is undecidable refer to the section 9.5 and 10.3.3.4.

Q.6 c) Refer to the section 10.3.2.2 for satisfiability problem and section 10.3.3.3 for Cook's theorem.