Solution 1:

Correct Answer: (C)

$S_1$ is false because it is possible to go back to some previous configuration and hence creating a cycle. $S_2$ is false because it is possible for some string to move to exactly one configuration from every configuration without going to any previous configuration and ultimately getting accepted or rejected and thus creating a line graph.

Solution 2:

Correct Answer: (D)

Counter example for $A$ and $C$ is 0011 getting rejected, for $B$ is 0010 getting rejected.

Now, let's see why $D$ is correct i.e. the language is accepting all and only palindromic strings. The idea is that TM start from the left most non-blank symbol of the input and after replacing it with a $\equiv$ the head moves to the right most non-blank symbol via $q_0$ and $q_c$, if the left most non-blank symbol was 0 or via $q_1$ and $q_{c_1}$ if the left most non-blank symbol was 1. There at $q_c$ or $q_{c_1}$ it checks if the left most non-blank symbol is same as the right most non-blank symbol or not. If they are not same then TM rejects input and if they are then it replaces the right most non-blank symbol with a $\equiv$ and again moves to the left most non-blank symbol to repeat the procedure. When the TM reaches the middle point then it simple moves to $q_{\text{accept}}$.

Solution 3:

Correct Answer: (D)

To prove that $S_1$ is correct we only need to show that a language $L$ accepted by a TM $M'$ in which we can stay at the current cell in addition to moving left and right at each step can also be accepted by a general TM $M$ as the other direction is trivially true. So, to get $M$ from $M'$ all we need to do is that for every transition of type $\delta(q,a) = (p,b,S)$ that is a transition where TM stayed at the current cell we change it to $\delta(q,a) = (r,b,L)$ and $\delta(r,c) = (p,c,R)$.

Again, to prove that $S_2$ is correct we only need to show that a language $L$ accepted by a TM $M'$ having a bidirectional tape can also be accepted a general TM $M$ as the other direction is trivially true. The basic idea is to fold the tape of $M'$ at the left most input symbol and use the alphabet $\Sigma^2$, where $\Sigma$ is the alphabet of $M'$. $M$ will use $M'$’s transition function but initially $M$ will read only the first symbol and ignore the second symbol in every cell. But when $M$ pass over the folding point, then instead it will start ignoring the first symbol in each cell and use only the second symbol and in this phase, it will translate left movements into right movements and vice versa. If it go over the folding point again, then it will again start reading the first symbol of each cell and translating movements normally.

Solution 4:

Correct Answer: (A,D)

Option $A$ is correct because NFA with even one queue is can simulate a TM (Remember lecture 3 of week 6). Thus the class of languages accepted by NFA with two queues is decidable languages and CFLs are proper subset of it.

Option $B$ is incorrect because priority queue can act like a normal queue, so every decidable language can be accepted by NFA with a priority queue, on the other hand a TM can simulate any model, so it can accept any language accepted by NFA with a priority queue. So both models have same class of language.
Option C is incorrect because DFA with 2 stacks accepts all and only decidable languages which is proper superset of DCFLs.

Option D is correct as it is similar to option A and we know that non-determinism does not change the power of TMs.

Solution 5:
Correct Answer: (B)

Counter example for option A and C is string 0101 which is not getting accepted, for option D it is string 001 which is not getting accepted.

Now, let's see why option B is correct. To check whether a string is of type $w0^n$ or not, the straight forward approach is to start from the left most symbol and then move to right most and see if it 0 or not, if it is 0 then remove the first and last symbol and perform the same operation on the remaining string. The TM given in the question basically does the same thing, it starts at the left most non-blank symbol with state $q_0$ and after replacing it with a blank it moves to the right most non-blank symbol at $q_2$. There if it sees a 1 or $\sqcup$ then it rejects that string else it replaces it with blank and move to the left most non-blank symbol at $q_0$ and repeat the same procedure again. Its easy to see now that strings accepted from this procedure are of the form mentioned in option B, thus B is the right answer.

Solution 6:
Correct Answer: (C)

Let $T_1$ be the TM for $L$ and $T_2$ be the TM for $M$.

We can make a TM $T_3$ for $Mix(L, M)$ as follows. Once the input $s$ is placed on input tape, $T_3$ will first check if the length of the input is even or not, if it is odd then it will reject it immediately else it will shuffle the tape content so that it looks like $a_1 a_2 \ldots a_k X b_1 b_2 \ldots b_k$ where $s = a_1 b_1 a_2 b_2 \ldots a_k b_k$. The set of states of $T_3$ is $Q \times P$ where $Q, P$ is the set of states of $T_1$ and $T_2$ respectively, in other words the states of $T_3$ are the pairs of states of $T_1$ and $T_2$. Now, the TM will start from $a_1$ and use $T_1$’s transition function to change the first element of the pair of states, change(or not) the content of $a_1$ and move the head appropriately. To remember the head position we can use a $\hat{\sqcup}$ symbol. Now it will cross $X$ and move to $b_1$ and simulate $T_2$. After that $T_3$ will keep moving between left hand side and right hand side and keep simulating the behaviour of $T_1$ and $T_2$. Finally the string will be accepted if both element of the state pair becomes accept state and rejected if any element of state pair becomes a reject state.

We can make a TM $T_4$ for $Perfectmix(L, M)$ using almost the similar idea using non-determinism. Once the input $s$ is placed on input tape, $T_4$ will first write a binary string of the length same as that of $s$ and then partition the string $s$ into two strings $s_1$ and $s_2$ such that $s_1$ is the concatenation of all the symbols of $s$ at whose index $T_4$ wrote 1 and $s_2$ is the concatenation of all the symbols of $s$ at whose index $T_4$ wrote 0. $T_4$ will then change the content of tape to $s_1 X s_2$. And then do the exact same thing as we did in case of $T_3$.

Solution 7:
Correct Answer: (B, C)

Option A is wrong because the machine doesn’t halt on string 11.
Option B is right because the machine halts on string 01.

Option C is right because the machine doesn’t halt if and only if it reaches \(q_2\), which it can reach only by reading at least two 1s. And the \(L((0 + 1)^{*}1(0 + 1)^{*}1(0 + 1)^{*})\) is actually the set of binary strings with at least two 1s.

Option D is wrong because the machine doesn’t halt on string 111.

Solution 8:

Correct Answer: (C)

Let \(M_1\) and \(M_2\) be TMs for \(L_1\) and \(L_2\).

First, let’s see why \(L_2 - L_1\) is Turing recognizable. Given an input the TM for \(L_2 - L_1\) will first run \(M_2\), if it rejects or doesn’t halt then clearly input does not belong to \(L_2\) and thus also does not belong to \(L_2 - L_1\), in case it accepts then we run \(M_1\) on given input and finally reject if \(M_1\) accepts it or else accept if \(M_1\) rejects it.

Now, consider the alphabet for \(L_1\) and \(L_2\) is \(\Sigma = \{0, 1\}\) and \(L_1 = \Sigma^*\), which is clearly decidable and \(L_2\) is some Turing recognizable language. Now \(L_1 - L_2\) in this case is the complement of \(L_2\) thus if \(L_1 - L_2\) is Turing recognizable that means that Turing recognizable languages are closed under complement, which we know is not true.

Thus option C is correct.