
CSL 705 Theory of Computation
Minor 1, Sem II 2010-11, Max 40, Time 1 hr

Name _____ Entry No. _____ Group _____

Note (i) Write your answers neatly and precisely in the space provided with each question including back of the sheet. You won't get a second chance to explain what you have written.

(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.

1. Consider a computing machine that is characterized by

- (i) a finite set of states Q
- (ii) two (infinite) stacks S_1, S_2
- (iii) a finite number of stack alphabet Γ with infinite supply of each alphabet
- (iv) transition function $\delta : Q \times \{\gamma_1, \gamma_2\} \rightarrow Q \times \{\gamma'_1, \gamma'_2\}$
where $\gamma_i \in \Gamma$, $\gamma'_i \in \Gamma \cup \epsilon$. Here γ_i are the symbols on the top of the two stacks S_1, S_2 and ϵ corresponds to the *pop* operation of the stack.

Show that this machine can simulate any Turing machine. **(8)**

The natural way to simulate a 1-tape Turing machine is to let the two stacks S_L and S_R contain the portion of the tape corresponding to the cells to the left of the head (inclusive) and the right of tape till the right-most cell that the tape head has visited (any cell to the right of this is surely blanks) - call this C_m . The bottommost symbol in S_L corresponds to the leftmost cell and the bottom-most symbol in S_R corresponds to C_m . For the next move of the TM, we need to maintain this invariant and update accordingly - this update can be done easily by examining the top of the left stack and by popping/pushing the right symbols as the head can move only one step at a time. (If you store the symbols in an inverted fashion then you need to go through many more steps).

Since the contents of the stack are in 1-1 correspondence with the TM tape, the simulation is faithful and a string is accepted by the stack machine when it reaches a final accepting state. Also note that any attempt to move the head left from the leftmost cell implies that we have emptied S_L and we can recognise this condition with a special bottom-stack symbol and the stack machine stops if an attempt is made to pop from an empty stack.

-
2. (i) If the complement of L is finite, is L recursive? Justify. (3)

Any finite set is recursive (simply enumerate the set), so complement is also recursive. (ii) If L is r.e., then is L^* r.e.? Recall that $w \in L^*$ iff $w = \epsilon$ or $w \in \underbrace{L \times L \times \cdots L}_i$ $i \geq 1$. (7)

Since $L^* = L^+ \cup \epsilon$ and ϵ is r.e., it suffices to show that L^+ is r.e. To see that, we must consider all non-empty partitions of the string w , say $w = w_1 w_2 \dots w_k, w_i \neq \epsilon$ and simulate the TM for L on w_1 followed by w_2 etc. If all of them are accepted then $w \in L^+$. However, the TM may not accept some string of a given partition, so we must *simultaneously* try all partitions. This could be in a round-robin fashion where we run the machine for i steps in the i -th round. Without this the proof doesn't work. An alternative (equivalent) is to use a ND TM and *guess* one of the right partitions.

Note that all partitions of a string of length is finite and can be generated by a TM.

3. Are the following problems decidable

- (a) Given a TM M and w , does M use finite number of tape cells for input w ? (7)

Let there be a TM M_F for this problem that halts for every input. We can show that $L_u \leq L_F$ in the following manner. First simulate M_F on $\langle M, w \rangle$. If it says no, when we answer no. Otherwise (i.e. it uses finite cells), we simulate M on w and keep track of the IDs that M goes through. Either M accepts w or some ID repeats indicating infinite loop, so we can decide if M accepts w or not given M_F .

Note that we have actually described a code of a machine that uses the code of M_F and a modified M_u (that stores all IDs of M).

Since L_u is undecidable, clearly M_F does not exist and hence undecidable.

- (b) Given a TM M with w as input does the head ever move left when started from the left-most square. (7)

We simulate the Turing machine for $|w| + |Q|$ steps. Either, it will move left in this period in which case we stop and say YES or it will never move left as some state will repeat after the head encounters the blank symbols at the end of w .

Without stating some finite bound by which we can decide the proof is not considered complete.

-
4. Let $L_1 \subset L_2 \subset L_3$. Can L_1 and L_3 be non r.e. and L_2 be recursive ? Justify. (8)

For simplicity consider unary languages and let $L' \subset 0^*$ be some non-r.e. language (we know they exist). Let $L_1 = 0^{2^i} | 0^i \in L'$. Clearly L_1 is also non-r.e. since $L' \leq L_1$. Let $L_2 = 0^{2^i}$ for all i and hence $L_1 \subset L_2$. Let $L'' = 0^{2^{i+1}} | 0^i \in L'$. Clearly L'' is also non r.e. Let $L_3 = L_2 \cup L''$.

Claim: L_3 is non r.e.

If not, we have a TM to recognise L'' in the following manner. For 0^j , if j is even, then it is not in L'' . Otherwise it is in L'' if it is in L_3 .