Question Paper for End Semester Examination
Course CS310

20 November, 2023

- The question paper carries 160 marks in total and consists of 9 questions.
- Justification is required for all questions unless explicitely stated as not required. Partial answers carry partial marks. Hence it is advantageous to show working of your answers.
- Illegible or un-understandable answers will get no marks. Pl. write clearly and accurately.
- This is a paper and pen examination. Answers must be written in an answersheet which must be submitted with roll number clearly marked.
- Additionally, Answer to each question must ALSO be uploaded on SAFE as an image answer.
- The examination must be completed in 150 minutes. Additional time will be given for SAFE upload. **Do not spend too much time on a single question.**
- Students may keep 3 printed or handwritten A4 size sheets with them for reference. The use of books, notebooks, laptops etc. is not allowed. **Mobile phones may be used ONLY FOR SAFE UPLOAD during the designated time.**
- Good Luck!

Note: $HP = \{\langle M\#x\rangle \mid M \text{ halts on } x\}$. Also, R.E. is the collection of recursively enumerable languages and Co-R.E is the collection of languages whose complement are in R.E.

Q1 (15 marks) For $\Sigma = \{a, b, c\}$ state whether the following statements are true or false. Justify your answers by a short proof outline or a counter-example, as appropriate.

(A) For any $A \subseteq \Sigma^*$ if A is regular then $L_1 = \{x \mid xx \in A\}$ is regular.

(B) For any $A \subseteq \Sigma^*$ if A is CFL then $L_2 = \{xx \mid x \in A\}$ is CFL.

(C) For any $A \subseteq \Sigma^*$ if A is CFL then $L_3 = \{x \mid xx \in A\}$ is CFL.

Solution:

(A) TRUE.

Let M DFA for A with set of states Q and transition function δ. We can can construct required NNFA M' for L_1. We have $Q' = Q^3$ with nondeterministic choice of starting states $\{(s,q,q) \mid q \in Q\}$. The idea is that for any word x we nondeterminstically guess start state (s,q,q) where $q = \hat{\delta}(s,x)$. The second compoent never changes. We have $\delta'((p,q,r),a) = (\delta(p,a), q, \delta(r,a))$. The set of accepting states is $\{(q,q,f) \mid q \in Q \wedge f \in F\}$.

(B) FALSE.

Choose $A = \Sigma^*$ then $L_2 = \{ww \mid w \in \Sigma^*\}$ which is known to be not CFL. L_2 can be shown to be not CFL using Pumping lemma for CFL. (Proof not required.)

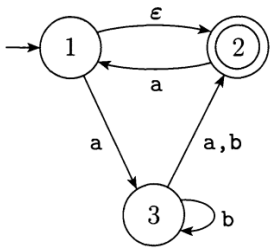
(C) FALSE.

Consider $A = \{a^m b^m c^n a^n b^p c^p \mid m,n,p \geq 0\}$. Then A is catenation of three CFLs and hence CFL. But for this A, we get $L_3 = \{a^n b^n c^n \mid n \geq 0\}$ as we force two halves to be same in A. This L_3 well-know to be non-CFL. This can be proved using Pumping Lemma for CFL. (Proof not required.)

Rubrics: For each of parts (A,B,C)

- Total 5 marks for correct option and explanation.
- 2 marks for correct option.
- 3 marks for correct justification.
- 1.5 marks for partially correct justification.

Q2 (24 marks) Consider the ϵ-NFA given below. The alphabet is $\{a,b\}$.



(A) Construct a DFA in transition table format for the above ϵ-NFA using the determinization method covered in class. The first row of the DFA transition table should be for the initial state. Show the complete calculation of the initial state of DFA as well as the calculations of table entries for the first row of the table. (For the remaining table entries, no calculations needs to be given but marks will be given only for correct entries).

(B) Minimize the DFA obtained in previous part using the Hopcroft partition refinement method. Indicate the phase at which each entry (i, j) enters the table. Also indicate a transition leading to this entry. How many states does your minimal automaton have?

(C) For the minimized automaton obtained in the previous part, construct a regular expression for the set of all words taking the initial state to itself. Follow the systematic procedure covered in Class and show the first step of recursive calculation. For subsequent steps, you may guess the required regular expressions without showing the working. No marks will be given for incorrect guesses.

Solution: (A) We construct the DFA using modified subset construction. States of DFA are subsets of the set of states of given NFA. We only retain reachable states in DFA. Let δ be the transition function of given NFA. Let δ' be the transition function of the constructed DFA.

- The start state $s = C_\epsilon(\{1\}) = \{1, 2\}$.
- $\hat{\delta}(\{1, 2\}, a) = \delta(1, a) \cup \delta(2, a) = \{1, 3\}$. Also, $C_\epsilon(\{1, 3\}) = \{1, 2, 3\}$. Hence $\delta'(\{1, 2\}, a) = \{1, 2, 3\}$.
- $\hat{\delta}(\{1, 2\}, b) = \delta(1, a) \cup \delta(2, a) = \emptyset$. Also, $C_\epsilon(\emptyset) = \emptyset$. Hence $\delta'(\{1, 2\}, b) = \emptyset$.

The transition table for constructed DFA is given below. * denotes final states.

	a	b
$\rightarrow \{1,2\}*$	$\{1,2,3\}$	$\emptyset$
$\{1,2,3\}*$	$\{1,2,3\}$	$\{2,3\}$
$\{2,3\}*$	$\{1,2\}$	$\{2,3\}$
$\emptyset$	$\emptyset$	$\emptyset$

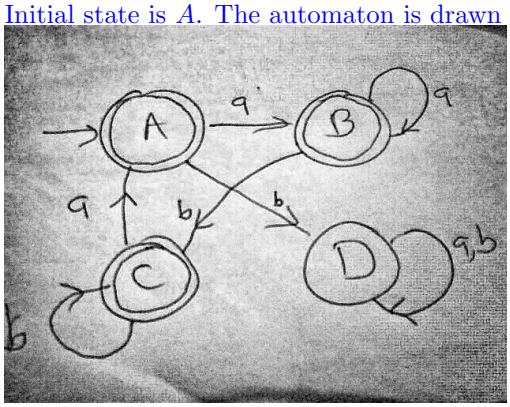
(B) We minimize the above automaton using Hopcroft partition refinement algorithm.

	$\{1,2\}$	$\{1,2,3\}$	$\{2,3\}$	$\emptyset$
$\{1,2\}*$	$\circ$			
$\{1,2,3\}*$	$\surd_1$	$\circ$		
$\{2,3\}*$	$\surd_1$	$\surd_2$	$\circ$	
$\emptyset$	$\surd_0$	$\surd_0$	$\surd_0$	$\circ$

Initially, in Phaze 0, we separate every final state from every non-final state. Next, we give a separating transition for each pair marked in each phase.
Phase 1: $\{1.2\}, \{1, 2, 3\} \xrightarrow{b} \emptyset, \{2, 3\}$. (tick) and $\{1.2\}, \{2, 3\} \xrightarrow{b} \emptyset, \{2, 3\}$. (tick). Note that neither a nor b separate $\{1, 2, 3\}, \{2, 3\}$ in phase 1.
Phase 2: $\{1, 2, 3\}, \{2, 3\} \xrightarrow{a} \{1, 2, 3\}, \{1, 2\}$ (tick)
Phase 3: No change.

All other states have their own equivalence class and the automaton is already Minimal. Minimal DFA has 4 states.

(C) We rename states as $\{1,2\} \to A$, $\{1,2,3\} \to B$, $\{2,3\} \to C$ and $\emptyset \to D$. Initial state is A. The automaton is drawn below.



Then,

$$\alpha_{A,A}^{A,B,C,D} \;=\; \alpha_{A,A}^{A,C,D} \;+\; \alpha_{A,B}^{A,C,D} \cdot (\alpha_{B,B}^{A,C,D})^* \cdot \alpha_{B,A}^{A,C,D}$$

Now by observation $\alpha_{A,A}^{A,C,D} = \epsilon$. Also, $\alpha_{A,B}^{A,C,D} = a$ and $\alpha_{B,B}^{A,C,D} = (\epsilon + a + baa)$ and $\alpha_{B,A}^{A,C,D} = ba$.
Hence the required regular expression is $\epsilon \;+\; a(\epsilon + a + baa)^*ba$.

Rubrics: 8 marks for each part. For A) 4 marks for correct table and 4 marks for correct explanation. For B) 6 marks for phases and transition , 2 marks for minimum number of states. For C) 2 marks for regular expression and 6 marks for the correct procedure.

Q3 (20 marks) Let $A/B = \{x \mid \exists y.\, xy \in A \wedge y \in B\}$ and let $A \leftarrow B = \{x \mid \forall y.\, (y \in B \Rightarrow xy \in A)\}$. Show that if A, B are regular then A/B and $A \leftarrow B$ are regular.

Solution: Given DFA $M_A = (Q_A, \Sigma, \delta_A, s_A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, s_B, F_B)$ for A and B, we define two computable predicates as follows. For each state $q \in Q_A$, let

- $M_q = (Q_A, \Sigma, \delta_A, q, F_A)$. Thus, M_q is same as DFA M_A but with start state changed to q.
- Let, M_1 be the product automaton $M_q \times M_B$ with set of final states $F_A \times F_B$. It recognizes the intersection of the two automaton languages. Let $Pred_1(M_A, M_B, q) = \top$ iff $L(M_1) \neq \emptyset$. We can use the emptiness checking decision procedure for DFA to compute this. Then, $Pred_1(M_A, M_B, q) = \top$ iff $\exists y.(\hat{\delta}_A(q, y) \in F_A) \wedge (\hat{\delta}_B(s_B, y) \in F_B)$

- Let M_2 be product automaton $M_q \times M_B$ with set of final states $((\sim F_A) \times F_B)$. It recognizes words y which are accepted by M_B but not accepted by M_q. Let $Pred_2(M_A, M_B, q) = \top$ iff $L(M_2) = \emptyset$. Then, $Pred_2(M_A, M_B, q) = \top$ iff $\forall y.(\hat{\delta}_B(q, y) \in F_B) \Rightarrow (\hat{\delta}_A(q, y) \in F_A)$. We can use the decision procedure for emptyness of DFA to compute this predicate.

We give an automaton N_1 for A/B Let $N_1 = (Q_A, \Sigma, \delta_A, F_1)$ where $F_1 = \{q \in Q_A \mid Pred_1(M_A, M_B, q)\}$. Then, N_1 accepts all x s.t $\hat{\delta}_A(s_A, x) = q$ and $\exists y.(\hat{\delta}_A(q, y) \in F_A) \wedge (\hat{\delta}_B(s_B, y) \in F_B)$. Hence, N_1 accepts A/B.

We given automaton N_2 for $A \leftarrow B$ Let $N_2 = (Q_A, \Sigma, \delta_A, F_2)$ where $F_2 = \{q \in Q_A \mid Pred_2(M_A, M_B, q)\}$. Then, N_2 accepts all x s.t $\hat{\delta}_A(s_A, x) = q$ and $\forall y.(\hat{\delta}_B(q, y) \in F_B) \Rightarrow (\hat{\delta}_A(q, y) \in F_A)$. Hence, N_2 accepts $A \leftarrow B$.

Rubrics: 10 marks for A/B and 10 marks for A$\leftarrow B$. About half marks have been given for A/B for automata accepting $\{xy \mid \exists y.\ xy \in A \wedge y \in B\}$. Some marks have also been given for effort..

Q4 (20 marks) For $w \in \{0,1\}^*$ let $\overline{w}$ be the bitwise complement of w. Thus, $\overline{110} = 001$. (Question is corrected as announced). Show that the complement $\sim L$ of language $L = \{w\overline{w} \mid w \in \{0,1\}^*\}$ is context-free, by constructing a PDA for $\sim L$. Explain the working of your PDA and how it accepts $\sim L$.

Solution Outline: Language $\sim L$ is disjoint union of (a) words 1of odd length and (b) words xy with the following property $|x| = |y|$ and there exists $0 < i < |x|$ s.t. $x[i] = y[i]$. Let $|x| = n$. Words of type (a) can be accepted by a DFA (and hence PDA). Below, we build a nondeterminstic PDA for the words of type (b). Union of the two PDAs will give the required PDA for $\sim L$.

In PDA for words of type (b), the stack is used as a counter. The PDA pushes letters $x[1], \ldots, x[i-1]$ on stack guessing i nondeterministically and it remembers the letter $x[i]$ in finite state control. In phase 2, it pushes letters $x[i+1], ..., x[n]$ on stack by guessing the start of y. In the next phase 3, it pops $i-1$ letters from the stack and also consumes $y[1], \ldots, y[i-1]$ by nondeterministically guessing position i in y. It then checks that $y[i]$ equals stored $x[i]$. It then pops remaining $n-i$ elements of the stack and consuming $y[i+1], \ldots, y[n]$. At the end the stack must be empty for accepting xy. It is easy to see that such nondeterminstic choice of position i in x, nondeterminstic choice of start of y and nondeterministic choice of i in y is possible iff $xy \in L$.

Rubrics:

2 marks for finding the correct complement of L10 marks for constructing the correct PDA 8 marks for explaining the working of the PDA and how it accepts the complement of L. Explanation must be given behind the idea and intuition

Q5 (20 marks) For $n \in N$ let $b(n)$ denote the binary representation of n without leading zeros, and $rev(x)$ denotes reverse of x. Thus $b(6) = 110$ and $rev(b(6)) = 011$. Let \$ be a symbol not in $\{0, 1\}$.

(A) Show that $L_1 = \{b(n)\$b(n+1) \mid n \geq 1\}$ is not CFL.

(B) Show that $L_2 = \{rev(b(n))\$b(n+1) \mid n \geq 1\}$ is CFL.

Solution Outline: (A) We use pumping lemma for CFL. Let Demon choose k. We choose word $z = 10^k01^k\$10^k10^k \in L_1$. We consider all possible split $z = uvwxy$ made by Demon subject to $|vwx| \leq k$ and $|vx| > 0$. Let $z' = uv^2wx^2y$.
Case 1: \$ occurs in v or x. Then z' will have more than one occurrences of \$ giving that $z' \notin L_1$.
Case 2: vwx occurs within the $b(n)$ (i.e. on left side of \$). Then in $z' = b(m)\$b(n+1)$, with $m \geq 2n$ as the $b(n)$ will have increased in length and $b(n)$ has leading 1. Hence $z' \notin L_1$.
Case 3: vwx occurs within the $b(n+1)$ (i.e. on right side of \$). The proof is similar to case 2.
Case 4: v occurs in $b(n)$ part and x occurs in $b(n+1)$ part. We will call the word to the left and right of \$ as lhs and rhs. Since $|vwx| \leq k$ we have $v = 1^p$. For any word in L if lhs has form 10^k01^* then rhs must be of the form 10^k10^* with exactly two occurrences of 1. Since $v = 1^p$, pumping up preserves the lhs form and hence rhs must also be of required form.
(Case 4.1) If $x = 10^q$ then pumping up will give z' where rhs will have more than two occurrences of 1. Hence $z' \notin L$.
(Case 4.2) $x = 0^q$ (as x must be close to v). We have $0 < p+q \leq k-2$. If $q = 0$ then, pumping up leaves rhs unchanged while changing lhs. Hence $z' \notin L$. If $q > 0$ then pumping makes rhs $10^{k+p}10^*$ which is not of the required form.

(B) We give a PDA recognizing L_2. The words in L_2 are either (a) of the form $1^k\$10^k$ or (b) of the form $1^k0\texttt{rev}(w)\$w10^k$. We construct two separate PDAs for words of type (a) and words of type (b). Then it is easy to recognize their union using non-determinstic choice of start transition.

For words of type (a) we pust all 1 on stack till \$. After \$, we consume 1 and then consume as many 0 as symbols on the stack (popping one 1 at a time while reading one 0). We accept by empty stack. On any mismatch we reject.

For words of type (b) we push the word before \$ on the stack. After \$ we match input w with reverse of w which is on the stack by poping the stack. If we find a 1 in input while top of stack has 0, we skip the input 1. After this we match every 0 in input with a 1 on stack. We accept by empty stack. On any mismatch we reject.

Rubrics: Part (A) - Total 11 marks

1. 2 marks for choice of word z in pumping lemma (there are other strings also that work, such as $1^k01^k\$1^{k+1}0^k$ - marking will be done similarly for these)

2. 1 mark for choice of pumping factor (for eg, taken as 2 in the solution)
3. 1 mark for case-1 ($ in v or x)
4. 3 marks for case-2 and case-3, where the marks are divided as 2+1 (i.e. only 2 marks if someone mentions only one of these 2 cases)
5. 2 marks for giving the possibilities on v and x in case-4
6. 1 mark each (1+1=2) for handling case-4.1 and case-4.2

A common mistake is to take the string as $1^k\$10^k$ when using pumping lemma. However, this fails for $u = 1^{k-1}, v = 1, w = \$1, x = 0, y = 0^{k-1}$. Any such wrong string will not attract any marks. Any other correct choice of string will be graded similarly to the rubrics above.

Part (B) - Total 9 marks

1. 2 marks for identifying the 2 kinds of strings that can come in L_2, and claiming that giving PDA separately for the two cases is enough
2. 3 marks for giving PDA for type (a)
3. 4 marks for giving PDA for type (b)

No partial marking will be done for each rubric of part (B). The pushes and pops have to be explained in the PDA, and both types of strings should be correctly identified. Any other complete and correct solution for (B) will also receive marks similarly (for example, another way to do this is to make $n+1$ on the stack using $\texttt{rev}(n)$, and then popping it out as $n+1$ comes). There are also some solutions that try to build a CFG for L_2, but their CFG accepts the language in which $b(n)$ can also start with leading zeros. If the overall CFG is correct but makes the above error, this will receive at most 3 marks.

If the overall solution is wrong, but the student has come up with some idea that is almost correct, then they can receive at most 1 mark for each part.

Q6 (10 marks) (A) Give a Turing machine accepting the language

$$L = \{\langle M \rangle \mid M \text{ accepts at least 221 strings}\}.$$

You may choose to describe your TM schematically at high level without giving its detailed description as a 9-tuple.
(B) Is L Recursive? Justify your answer with a proof.

Solution: (A) We design a TM N for L. Strategy: Given input $\langle M \rangle$, to determine whether $M \in L$, TM N can systematically enumerate all $(x, n) \in \Sigma^* \times N$. It can simulate M on x for n steps. Every time, we find a new x which accepts in some n steps, we add it to a list of accepted words and also increment the count of this list. If list size becomes 221 then N halts and ACCEPTs. Otherwise the search of x continues with next enumeration. It is

clear that if $|L(M)| \geq 221$ then N will accept M and otherwise it will loop forever. Thus N accepts L. This shows that L is R.E.

(B) We will show that $HP \leq_m L$. Since HP is not RECURSIVE, this shows that L is not RECURSIVE.

We define $\sigma(\langle M\#x\rangle) = \langle N\rangle$ where TM N on in input y behaves as follows:

- Erase y from input tape WT1
- Write x on WT1
- Simulate M on x using WT1
- If M halts then ACCEPT y.

If M halts on x then $L(N) = \Sigma^*$ giving $N \in L$.
If M does not halt on x then $L(N) = \emptyset$ giving $N \notin L$.
Hence $\sigma : HP \leq_m L$.

Rubrics:

- 3 marks for the clear explanation that TM halts when it finds 221 strings.
- 2 marks for correctly interpreting that it will loop forever otherwise.
- 2 marks for correctly interpreting that L is not recursive.
- 3 marks for correct proof.

Q7 (8 marks) Let $L = \{\langle M\rangle \mid M$ halts on all inputs in less than 221 steps$\}$ State whether L is R.E. Is it Co-R.E? Justify your answers.

Solution: Given $M\#x$ if it halts in less than 221 steps then it will only be able to look at atmost initial 221 input symbols in x, denoted $x[1:221]$. The idea is that we can enumerate all possible inputs y of length upto 221 and simulate M for upto 221 steps. If an input y is found where M does not halt then $M \notin L$. If no such y is found then $M \in L$. We can design a TTM N accepting L.

Turing machine N on input $\langle M\rangle$ behaves as follows:

- Systematically enumerate all $y \in \Sigma^*$ with $|y| \leq 221$. This can be done in increasing order of $|y|$. This is a finite list.
- For each y simulate M for 221 steps.
- If M does not halt within 221 steps then N REJECTS M.
- If M halts within 221 steps go to next enumeration of y.
- If enumeration of all y finishes then N ACCEPTS M.

By the previous discussion, it is clear that $L(N) = L$.

Note that N is a total Turing machine. Hence L is RECURSIVE giving that it is both R.E. and Co-R.E.

Rubrics:

- 2 Marks for idea of enumerating all size inputs upto 221
- 1 Mark stating the Idea of TTM
- 3 Marks for correct working of TTM
- 2 Marks for stating it recursive or R.E. and Co-R.E.

Q8 (20 marks) Let $HALTFIN = \{\langle M\rangle \mid M \text{ halts on finitely many inputs}\}$ be a language of TM codes.
(A) Prove that $HALTFIN$ is not R.E. as well as its complement $\sim HALTFIN$ is also not R.E. You may assume that HP is R.E. but not RECURSIVE and use many-to-one reductions if required. Do not assume anything else.
(B) Can you use Rice's theorem to prove the above result? Pl. explain why or why not?

Solution: (A) We will show that $HP \leq_m HALTFIN$ as well as $\sim HP \leq_m HALTFIN$. The former implies that $\sim HP \leq_m \sim HALTFIN$. Since $\sim HP$ is not R.E. we have that neither $HALTFIN$ nor $\sim HALTFIN$ are R.E.

(1) Let $\sigma_1(\langle M\#x\rangle) = N$ where N on input y behaves as follows.

- N erases y on input tape.
- N writes x on input tape.
- N simulates M on x using input tape.
- N Accepts if M halts.

If M halts on x then N accepts every y. Hence $N \notin HALTFIN$.
If M does not halt on x then N loops on every y. Hence $N \in HALTFIN$.
This shows that $\sigma_1 :\sim HP \leq_m HALTFIN$

(2) Let $\sigma_2(\langle M\#x\rangle) = N$ where N on input y behaves as follows.

- N gets y on input tape.
- N writes x on WT1.
- N simulates M on x on WT1 for $|y|$ steps.
  This is done by erasing one symbol of y in each move.
- If M halts on x before y is completely erased then N Loops
- If M does not halts on x when y is completely erased then N Accepts

If M does not halt on x then for all y we have N halts and accepts. Hence $\langle N\rangle \notin HALTFIN$.
If M halts on x in say n steps, then (a) for all y with $|y| \leq n$ TM N halts and Accepts. (b) Moreover, for all y with $|y| > n$ TM N loops. Hence, $\langle N\rangle \in HALTFIN$.
This shows that $\sigma_2 : HP \leq_m HALTFIN$

(B) Rices Theorem Rice's theorem can only be used for properties of R.E. sets which are properties of language of the TM. There is M_{rej} which halts and rejects all words. There is TM M_{loop} which loops on all words. We have $L(M_{rej}) = L(M_{loop}) = \emptyset$. But $M_{loop} \in HALTFIN$ whereas $M_{rej} \notin HALTFIN$. Hence, $HALTFIN$ is not a property of R.E. sets and Rice's theorem cannot be applied to it.

Rubrics:

- (A) 1 mark for $\sim HP$ is not R.E.
- (A) 2 marks for stating $\sim HP \leq_m \sim HALTFIN$
- (A) 2 marks for stating $\sim HP \leq_m HALTFIN$
- (A) 6 marks for proving $HP \leq_m HALTFIN$ (or equivalent). Divided as 4 for correct behaviour of N and 2 for showing the reduction
- (A) 4 marks for proving $\sim HP \leq_m HALTFIN$ (or equivalent). Divided as 3 for correct behaviour of N and 1 for showing the reduction
- (B) 1 marks for stating not possible as theorem is for R.E. sets
- (B) 2 marks for stating $HALTFIN$ is not property of R.E. sets. and 2 marks for proving $HALTFIN$ is not property of R.E. sets

Q9 (23 marks) Let a Bi-Property $PROP(\langle M_1 \# M_2 \rangle)$ be a property which given the codes of two TMs defines whether the property holds or not. E.g. $PROP_1(\langle M_1 \# M_2 \rangle)$ holds iff $L(M_1) = L(M_2)$.

(A) State an extention of Rice's Theorem (both part) to bi-properties which gives sufficient condition to check that the given bi-property $PROP$ is not RECURSIVE, or not R.E., respectively. Clearly state the conditions on $PROP$ for applying the extended theorem.

(B) Give a proof of the extended Rice's theorem (both parts).

(C) Using the above Theorem, can you conclude that $PROP_1$ defined above is not RECURSIVE? Can you conclude that $PROP_1$ is not R.E.? Pl. justify.

Solution: (A) Define $PROP$ to be bi-property of R.E. sets if $L(M_1) = L(M_1') \wedge L(M_2) = L(M_2')$ implies $PROP(\langle M_1 \# M_2 \rangle) = PROP(\langle M_1' \# M_2' \rangle)$. Also, $PROP$ is called non-trivial if there exist TMs M_0, M_1, M_2, M_3 s.t. $PROP(\langle M_0 \# M_1 \rangle) = \top$ and $PROP(\langle M_2 \# M_3 \rangle) = \bot$. Finally, $PROP$ is called non-monotonic if there exist TMs M_0, M_1, M_2, M_3 s.t.
(**) $PROP(M_0, M_1) = \top$ and $PROP(M_2, M_3) = \bot$ and $L(M_0) \subseteq L(M_2)$ and $L(M_1) \subseteq L(M_3)$.

Extended Rice's Theorem (Part 1): Any non-trivial bi-property of R.E. sets is not RECURSIVE (i.e. not DECIDABLE).
Extended Rice's Theorem (Part II): Any non-monotonic bi-property of R.E. sets is not R.E..

(B) Proof of Theorem Part 1 : Given $\langle M\#x\rangle$ and TM N let $Comp_1(M\#x,\ N)$ be a TM which

- Takes input y on WT0
- Writes x on WT1
- Simulates M on x on WT1
- If M halts on x then Simulates N on y on WT0.
- Accepts if N accepts and Rejects if N rejects.

Then, if M halts on x then $L(Comp_1(M\#x,\ N)) = L(N)$, and if M does not halt on x then $L(Comp_1(M\#x,\ N)) = \emptyset$.

Let M_{rej} be TM which rejects every word.
Also, let $HAS(PROP) = \{\langle M_0\#M_1\rangle \mid PROP(\langle M_0\#M_1\rangle) = \top\}$.
Case 1: $PROP(\langle M_{rej}\#M_{rej}\rangle) = \bot$.
As $PROP$ is non-trivial, there exist M_0, M_1 s.t. $PROP(\langle M_0\#M_1\rangle) = \top$.
Define many-to-one reduction $\sigma : HP \leq_m HAS(PROP)$ as follows: $\sigma(\langle M\#x\rangle) = (Comp_1(M\#x,\ M_0)\#Comp_1(M\#x, M_1))$.

- If M halts on x then $L(Comp(M\#x,\ M_0)) = L(M_0)$ and $L(Comp(M\#x,\ M_1)) = L(M_1)$ giving $PROP(\sigma(\langle M\#x\rangle)) = \top$. This is because property $PROP$ depends only on language and not on TM structure.
- If M does not halts on x then $L(Comp(M\#x,\ M_0)) = L(M_{rej})$ and $L(Comp_1(M\#x \downarrow M_1)) = L(M_{rej})$ giving $PROP(\sigma(\langle M\#x\rangle)) = \bot$.
- Hence we have $\sigma : HP \leq_m HAS(PROP)$.
- Since HP is not RECURSIVE we get that $HAS(PROP)$ is not RECURSIVE and hence $PROP$ is not DECIDABLE.

Case 2: $PROP(\langle M_{rej}\#M_{rej}\rangle) = \top$.
There exist M_2, M_3 s.t. $PROP(\langle M_2\#M_3\rangle) = \bot$.
Define $\sigma(\langle M\#x\rangle) = Comp_1(M\#x,\ M_2)\#Comp_1(M\#x,\ M_3)$. Then a similar argument to above shows that $\sigma :\sim HP \leq_m HAS(PROP)$. Since, $\sim HP$ not R.E., we can conclude that $HAS(PROP)$ is not R.E. and hence $PROP$ is not decidable.

(B) Proof of Theorem Part II Given $\langle M\#x\rangle$ and TMs N_1, N_2 with $L(N_1) \subseteq L(N_2)$ we construct TM $Comp_2(M\#x,\ N_1,\ N_2)$ which behaves as follows.

- Takes input y on WT0
- Copies input y from WT0 to WT1
- Writes x on WT2
- It simulates in parallel three Turing machines on three tapes doing one step of each.
  - Simulates M on x on WT2
  - Simulates N_1 on y on WT0
  - Simulates N_2 on y on WT1
- It ACCEPTS on the first occurrence of any of the following conditions:
  - N_1 accepts
  - M halts on x and N_2 accepts.

Then, if M halts on x then $L(Comp_2(M\#x,\ N_1,\ N_2)) = L(N_2)$, and
if M does not halt on x then $L(Comp_2(M\#x,\ N_1,\ N_2) = \emptyset$.

If $PROP$ is non-monotonic, there exist M_0, M_1, M_2, M_3 with properties ** stated in the definition of non-monotonicity.
We define $\sigma_2(M\#x) = Comp_2(M\#x,\ M_0,\ M_2)\#Comp_2(M\#x,\ M_1,\ M_3)$.
Note that this is well-formed.

If M halts on x then $L(Comp_2(M\#x,\ M_0,\ M_2)) = L(M_2)$ and $L(Comp_2(M\#x,\ M_1,\ M_3)) = L(M_3)$. Hence $PROP(\sigma_2(M\#x) = \bot$.
If M does not halt on x then $L(Comp_2(M\#x,\ M_0,\ M_2)) = L(M_0)$ and $L(Comp_2(M\#x,\ M_1,\ M_3)) = L(M_1)$. Hence $PROP(\sigma_2(M\#x) = \top$.
Hence, $\sigma_2 :\ \sim HP \leq_m HAS(PROP)$. Since $\sim HP$ is not $R.E.$ we have that $HAS(PROP)$ is not R.E. giving that $PROP$ is not R.E.

(C) $PROP_1$ is not R.E. and hence Not DECIDABLE . The proof is as follows: Note that $PROP_1$ is a bi-property of R.E. sets as it is defined in terms of languages.
Also, $L(M_{rej}) = L(M_{rej})$ giving $PROP_1(M_{rej}\#M_{rej}) = \top$. Also, $L(M_{rej}) = \emptyset \subset \Sigma^* = L(M_{acc})$ giving $PROP_1(M_{rej}\#M_{acc}) = \bot$. Hence $PROP_1$ is non-monotonic.
By Extended Rices Theorem Part II, we get that $PROP_1$ is not R.E. and hence not DECIDABLE.

Rubrics:

- Totally 6 Marks for the correct statement of Rices Theorems (Part 1 and II). This is divided as follows: 1 Mark for requiring property of R.E. sets. 2 Marks for non-trivial property in Part 1 and 2 marks for non-monotonicity in Part II.

- 7 Marks for the proof of Theorem Part 1
- 8 Marks for the proof of Theorem Part II.
- 2 Marks for showing that $PROP_1$ is not R.E. Give only 1 Mark if $PROP_1$ is shown to be not RECURSIVE only.