UMC205

Assignment 05 – ⟨21·04·2025⟩

AUTOMATA THEORY AND COMPUTABILITY

Topics – ⟨Turing Machines And Decidability⟩

Name → Piyush Kumar
Sr./No. → 23801

Problem 5.1. Is the following question decidable: Given a Turing machine M and a state q of M, does M ever enter state q on some input? Justify your Answer.

Solution: No., Because The halting problem can be reduced to this. Given any Turing machine M and input x, we can construct a new Turing Machine $P_{M,x}$ that erases its input and runs M on x. Thus $P_{M,x}$ halts on all inputs iff M halts on x. Let t be the accept state of $P_{M,x}$.

This map $M\#x \mapsto P_{M,x}\#t$ is computable, while gives a reduction from HP to $\{M\#q \mid M \text{ enters state } q\}$.

Thus, if it were decidable whether $P_{M,x}$ enters state t on some input, we could decide the halting problem.

Problem 5.2. An enumeration machine N is a two-tape Turing machine with the following distinctions:

- The machine is not given any input; both its tapes are blank to begin with.
- The first tape is a write only tape, on which the machine can only write symbols of Σ. The second tape is a usual two-way read/write tape on which it can write any element of Γ.
- The machine has no accept/reject state, but instead it has a special enumeration state 'e' by which it signals that it has written something interesting on its first tape. Thus the contents of the first tape are said to have been "enumerated" whenever the machine enters the state 'e'. After entering e, the contents of the first tape are "automatically" erased and the first tape are ~~[illegible]~~ head is rewound to the left end of the tape. The machine then resumes working from there.
- The language L(N) is defined to be the set of strings in Σ^* enumerated by N.

(a) Prove that enumeration machines and Turing machines are equal in a computation power: i.e. the class of languages they define is precisely the r.e. languages.

Solution: We claim: The class of languages defined by enumeration machines is precisely the class of (r.e.) languages.

Proof: ($\Rightarrow$) Every r.e. language can be enumerated by some enumeration machine.

Let us suppose L be an r.e. language. Then there exists a Turing machine M such that $L = L(M)$, and M accepts all strings in L (but may not hault on strings not in L). we construct an enumeration machine N as follow:

- N systematically enumeration all strings $w_1, w_2, w_3, \dots, \Sigma^*$
- It simulates M on all w_i using dovetailing.

- Whenever $M(w_i)$ accepts, N writes w_i on the write-only and enters the enumerations state e.

Therefore, N enumerates all and only strings in L, so $L(N)=L$.

(⇐) Every language enumerated by an enumeration machine is r.e. Let us suppose N be an enumeration machine. Define $L=L(N)$ to be set of the strings it enumerates.

To construct a Turing machine M that recognizes L, simulate N by step. Each time N enters the enumeration state e, compare the output string with the input w. If the output equals w, then M accepts. Hence, M recognizes L, and L is recursively enumerable.

(b) Prove that an r.e. language is recursive iff there is an enumeration machine that enumerates it in increasing order.

Proof: (⇒) If L is recursive, then it can be enumerated in increasing order.

Let 'L' be recursive. Then there exists a Turing machine M that decides L. Then we construct an enumeration machine N as follows:

- Enumerate all strings $w_1, w_2, w_3, \dots$ in Σ^* in increasing order.
- For each w_i, simulate $M(w_i)$. If $M(w_i)$ accepts, write w_i on the write only tape and enter state e.

Since M haults on all inputs, N will enumerated exactly the strings in L, and in increasing order.

(⇐) If L is r.e. and can be enumerated in increasing order, then L is recursive.

Suppose N enumerates L in increasing order.

To decide whether a string w belongs to L, simulate N:

- Every time N enters e, compare the enumerated string with w.
- If w is enumerated, accept.
- If a string greater than w is enumerated before w, reject Since L is ordered, w cannot appear after)

Hence, L is decidable and therefore recursive.

Conclusion that An r.e. language is recursive if and only if there exist an enumeration machine that enumerates it in increasing order.

▨

Problem 5.3. Show that neither the language

$$TOTAL = \{M \mid M \text{ halts on all inputs}\}$$

nor its complement is r.e.

Solution: Both proofs are by reducing $\neg HP$ to the given language. The reduction for $\neg$ Total is easier, so we present the proof first.

$\neg HP \leq \neg TOTAL$. Given a Turing machine M and input x, we can construct a new Turing Machine $P_{M,x}$ that erases its input and runs M on x. Thus if M halts on x, then $P_{M,x}$ halts on all inputs. If M does not halt on x, then $P_{M,x}$ does not halt on any input.

Let φ be the map that sends $M \# x$ to $P_{M,x}$. Then φ is a computable function, and

$$M \# x \in \neg HP \Leftrightarrow \varphi(M \# x) \in \neg TOTAL.$$

This proves the claim.

The reduction for TOTAL is more involved.

TOTAL. Given a Turing machine M and input x, we can construct a new Turing machine $Q_{M,x}$ that does the following:

(i) For input w, simulate M on x for $|w|$ steps.

(ii) If M halts on x within $|w|$ steps, then enter a looping state.

(iii) If M does not halt on x within $|w|$ steps, then halt.

This is easy to achieve by storing the input w on one tape, and simulating M on x on another tape, blanking one letter from w after each step of the simulation. This is a computable function.

If M halts on x in n steps, then $Q_{M,x}$ enters an infinite loop for inputs longer than n. But if M does not halt on x, then $Q_{M,x}$ halts on every input. Thus

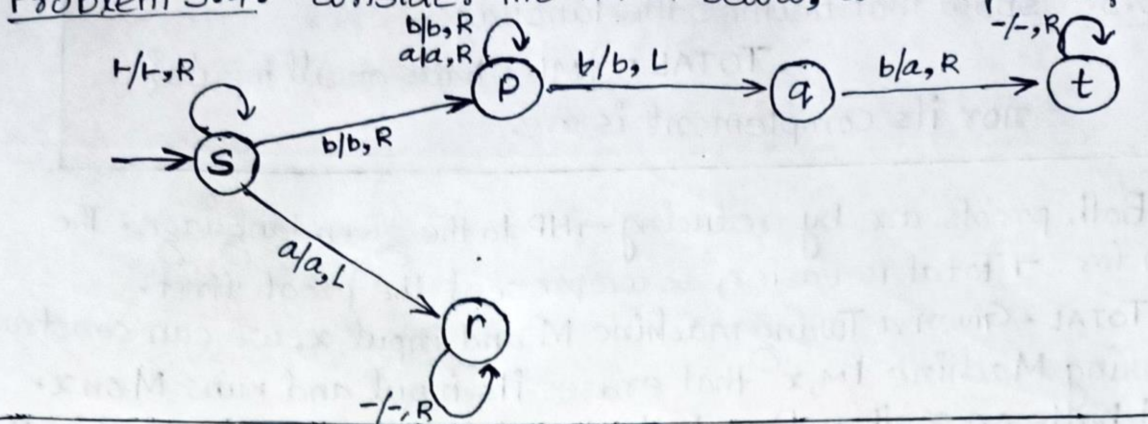
$$M \# x \in \neg HP \Longleftrightarrow Q_{M,x} \in TOTAL.$$

This proves the claim.

Since $\neg HP$ is not recursively enumerable, neither TOTAL nor its complement is recursively enumerable.

∎

Problem 5.4. Consider the TM M below, with input alphabet $\{a, b\}$



(a) Give any string in Valcomp$_{M}$, baabb.

Solution:- $\underset{s}{\vdash}\,\underset{-}{b}\,\underset{-}{a}\,\underset{-}{a}\,\underset{-}{b}\,\underset{-}{b}\,\underset{-}{\flat}\,\#\,\underset{-}{\vdash}\,\underset{s}{b}\,\underset{-}{a}\,\underset{-}{a}\,\underset{-}{b}\,\underset{-}{b}\,\underset{-}{\flat}\,\#\,\underset{-}{\vdash}\,\underset{-}{b}\,\underset{p}{a}\,\underset{-}{a}\,\underset{-}{b}\,\underset{-}{b}\,\underset{-}{\flat}\,\#$

$\underset{-}{\vdash}\,\underset{-}{b}\,\underset{-}{a}\,\underset{p}{a}\,\underset{-}{b}\,\underset{-}{b}\,\underset{-}{\flat}\,\#\,\underset{-}{\vdash}\,\underset{-}{b}\,\underset{-}{a}\,\underset{-}{a}\,\underset{p}{b}\,\underset{-}{b}\,\underset{-}{\flat}\,\#\,\underset{-}{\vdash}\,\underset{-}{b}\,\underset{-}{a}\,\underset{-}{a}\,\underset{-}{b}\,\underset{p}{b}\,\underset{-}{\flat}\,\#$

$\underset{-}{\vdash}\,\underset{-}{b}\,\underset{-}{a}\,\underset{-}{a}\,\underset{-}{b}\,\underset{-}{b}\,\underset{p}{\flat}\,\#\,\underset{-}{\vdash}\,\underset{-}{b}\,\underset{-}{a}\,\underset{-}{a}\,\underset{-}{b}\,\underset{q}{b}\,\underset{-}{c}\,\#\,\underset{-}{\vdash}\,\underset{-}{b}\,\underset{-}{a}\,\underset{-}{a}\,\underset{t}{b}\,\underset{-}{c}\,\underset{-}{c}\,\#$

(b) Recall the notion of matching triples of symbols used in class. Give the entire set of matching triples for M.

Solution: We assume the unspecified transitions to be –/–, R into the reject state r. ~~as discussed in~~ We split the matching triples by state. we let x, y, z denote any tape symbol, and β denote any non-blank symbol. Also let B denote any symbol that is not b. We denote a matching triple as ⟨triple1 | triple2⟩ for clarity.

The matching triples for s are

$\left\langle \begin{matrix} \vdash & x & y \\ s & - & - \end{matrix} \,\middle|\, \begin{matrix} \vdash & x & y \\ - & s & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & \vdash & y \\ - & s & - \end{matrix} \,\middle|\, \begin{matrix} x & \vdash & y \\ - & - & s \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & y & \vdash \\ - & - & s \end{matrix} \,\middle|\, \begin{matrix} x & y & \vdash \\ - & - & - \end{matrix} \right\rangle$

$\left\langle \begin{matrix} a & x & y \\ s & - & - \end{matrix} \,\middle|\, \begin{matrix} a & x & y \\ - & - & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & a & y \\ - & s & - \end{matrix} \,\middle|\, \begin{matrix} x & a & y \\ r & - & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & y & a \\ - & - & s \end{matrix} \,\middle|\, \begin{matrix} x & y & a \\ - & r & - \end{matrix} \right\rangle$

$\left\langle \begin{matrix} b & x & y \\ s & - & - \end{matrix} \,\middle|\, \begin{matrix} b & x & y \\ - & p & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & b & y \\ - & s & - \end{matrix} \,\middle|\, \begin{matrix} x & b & y \\ - & - & p \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & y & b \\ - & - & s \end{matrix} \,\middle|\, \begin{matrix} x & y & b \\ - & - & - \end{matrix} \right\rangle$

$\left\langle \begin{matrix} c & x & y \\ s & - & - \end{matrix} \,\middle|\, \begin{matrix} c & x & y \\ - & r & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & c & y \\ - & s & - \end{matrix} \,\middle|\, \begin{matrix} x & c & y \\ - & - & r \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & y & c \\ - & - & s \end{matrix} \,\middle|\, \begin{matrix} x & y & c \\ - & - & - \end{matrix} \right\rangle$

$\left\langle \begin{matrix} \flat & x & y \\ s & - & - \end{matrix} \,\middle|\, \begin{matrix} \flat & x & y \\ - & r & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & \flat & y \\ - & s & - \end{matrix} \,\middle|\, \begin{matrix} x & \flat & y \\ - & - & r \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & y & \flat \\ - & - & s \end{matrix} \,\middle|\, \begin{matrix} x & y & \flat \\ - & - & - \end{matrix} \right\rangle$

For p, the matching triples are

$\left\langle \begin{matrix} \flat & x & y \\ p & - & - \end{matrix} \,\middle|\, \begin{matrix} c & x & y \\ - & - & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & \flat & y \\ - & p & - \end{matrix} \,\middle|\, \begin{matrix} x & c & y \\ q & - & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & y & \flat \\ - & - & p \end{matrix} \,\middle|\, \begin{matrix} x & y & c \\ - & q & - \end{matrix} \right\rangle$

$\left\langle \begin{matrix} \beta & x & y \\ p & - & - \end{matrix} \,\middle|\, \begin{matrix} \beta & x & y \\ - & p & - \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & \beta & y \\ - & p & - \end{matrix} \,\middle|\, \begin{matrix} x & \beta & y \\ - & - & p \end{matrix} \right\rangle$ $\left\langle \begin{matrix} x & y & \beta \\ - & - & p \end{matrix} \,\middle|\, \begin{matrix} x & y & \beta \\ - & - & - \end{matrix} \right\rangle$

For q, they are;

$$\left\langle \begin{matrix} b & x & y \\ q & - & - \end{matrix} \middle| \begin{matrix} c & x & y \\ - & - & - \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & b & y \\ - & q & - \end{matrix} \middle| \begin{matrix} x & c & y \\ t & - & - \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & y & b \\ - & - & q \end{matrix} \middle| \begin{matrix} x & y & c \\ - & t & - \end{matrix} \right\rangle$$

$$\left\langle \begin{matrix} B & x & y \\ q & - & - \end{matrix} \middle| \begin{matrix} B & x & y \\ - & r & - \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & B & y \\ - & q & - \end{matrix} \middle| \begin{matrix} x & B & y \\ - & - & r \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & y & B \\ - & - & q \end{matrix} \middle| \begin{matrix} x & y & B \\ - & - & - \end{matrix} \right\rangle$$

For r, they are

$$\left\langle \begin{matrix} x & y & z \\ r & - & - \end{matrix} \middle| \begin{matrix} x & y & z \\ - & r & - \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & y & z \\ - & r & - \end{matrix} \middle| \begin{matrix} x & y & z \\ - & - & r \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & y & z \\ - & - & r \end{matrix} \middle| \begin{matrix} x & y & z \\ - & - & - \end{matrix} \right\rangle$$

The same triples apply for the accept state t, with r replaced by t.

$$\left\langle \begin{matrix} x & y & z \\ t & - & - \end{matrix} \middle| \begin{matrix} x & y & z \\ - & t & - \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & y & z \\ - & t & - \end{matrix} \middle| \begin{matrix} x & y & z \\ - & - & t \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & y & z \\ - & - & t \end{matrix} \middle| \begin{matrix} x & y & z \\ - & - & - \end{matrix} \right\rangle$$

In addition to these, we have the matching triples

$$\left\langle \begin{matrix} x & y & z \\ - & - & - \end{matrix} \middle| \begin{matrix} x & y & z \\ \theta & - & - \end{matrix} \right\rangle \quad \left\langle \begin{matrix} x & y & z \\ - & - & - \end{matrix} \middle| \begin{matrix} x & y & z \\ - & - & \theta \end{matrix} \right\rangle$$

For every state θ, since the read head could be position right outside the triple and move in. Lastly, we have the matching triples

$$\left\langle \begin{matrix} x & y & z \\ - & - & - \end{matrix} \middle| \begin{matrix} x & y & z \\ - & - & - \end{matrix} \right\rangle$$

Where the read head is not in the triple either before or after the transition.

(C) Justify the claim that for two valid configurations C_1 and C_2 of M, which are the same length, we have: $C_1 \xRightarrow{1} C_2$ iff for each position in C_1, the triple of symbols in C_1 and the corresponding triple in C_2 match.

Solution: The construction of the triples makes it clear that if $C_1 \xRightarrow{1} C_2$, then each triple in C_1 matches the corresponding triple in C_2.
The converse is also true. Since C_1 and C_2 are valid configurations, they have exactly one state symbol in the bottom row. The triples matching ensures that the state symbol moves at most one position, either left or right. Thus there are at most 6 triples which contain any state symbol (before or after the transition). The matching triples ensure that these have arisen from valid transitions.

For the other triples, the matching ensures that no tape symbol has changed, since the only matching triples with no state symbol is

$$\left\langle \begin{matrix} x & y & z \\ - & - & - \end{matrix} \middle| \begin{matrix} x & y & z \\ - & - & - \end{matrix} \right\rangle.$$

Notice, that having some length greater than 1 is essential. Consider the transition

$$\begin{pmatrix} u & v & w & x & y & z \\ - & - & q & - & - & - \end{pmatrix} \xRightarrow{1} \begin{pmatrix} u & v & w' & x & y & z \\ - & - & - & q & - & - \end{pmatrix}$$ The match

$$\left\langle \begin{matrix} u & v & w \\ - & - & q \end{matrix} \middle| \begin{matrix} u & v & w \\ - & - & - \end{matrix} \right\rangle$$

ensure that the letter change and move are consistent. But what for the matching triple

$$\left\langle \begin{matrix} x & y & z \\ - & - & - \end{matrix} \;\middle|\; \begin{matrix} x & y & z \\ q & - & - \end{matrix} \right\rangle ?$$

The state could 'vanish' in the false transition

$$\begin{pmatrix} u & v & w & x & y & z \\ - & - & q & - & - & - \end{pmatrix} \xRightarrow{1} \begin{pmatrix} u & v & w' & x & y & z \\ - & - & - & - & - & - \end{pmatrix}$$

and both the initial and final triples would match. But the triples in between reject this possibility, since

$$\left\langle \begin{matrix} v & w & x \\ - & q & - \end{matrix} \;\middle|\; \begin{matrix} v & w' & \\ x & - & - \end{matrix} \right\rangle$$

is never a matching triple. The length provides immediate context to each position. This is more clear from the next part.

<u>Note that</u>:

The desirable properties still hold. If C_1 and C_2 are valid configurations of the same length, then $C_1 \xRightarrow{1} C_2$ iff for each position in C_1, the pair of symbols in C_1 and the corresponding pair in C_2 match.
This is because for any transition which changes a portion of the tape as

$$\begin{pmatrix} u & v & w & x & y \\ - & - & q & - & - \end{pmatrix} \xRightarrow{1} \begin{pmatrix} u & v & w' & x & y \\ - & - & - & q & - \end{pmatrix}$$

the pair at w ensures that the head has moved right by following the correct transition. The pair at v ensures that the head has not moved left.

Problem 5.5. Is it decidable whether the complement of a given CFL is a CFL? Justify your Answer.

Solution: We will reduce the halting problem to this problem. Let M is the Turing machine and x be an input. Define two PDAs M_1 and M_2 as in class: Given a string $c_0 \# c_1 \# \ldots \# c_n \#$ of (reversed) Configurations,

(i) M_1 checks that even-numbered configurations are correctly followed by their successor.

(ii) M_2 checks that the odd-numbered configurations are correctly followed by their successor.

Then $L(M_1) \cap L(M_2)$ is the set of valid computations of M on x. This is precisely $\mathrm{Valcomp}_{M,x}$. If x does not halts on M, this is empty, and hence a CFL. If x halts on M, this is not a CFL.

Thus, the map $M\#x \longmapsto (M_1, M_2)$ is a reduction from the halting problem to the problem of determining whether the intersection of two given CFLs is a CFL. ▨