Assignment 5

Naman Mishra
(22223)

13 April, 2024

Problem 5.1. Show that the function $square\colon \mathbb{N} \to \mathbb{N}$, given by $square(n) = n^2$ is computable by a Turing machine in the sense discussed in class. Give a complete description of the moves of the TM in a modular way.

Solution. A Turing machine for this computation is shown in figure 1. That is, a machine over the input alphabet $\{0\}$ that inputs 0^n and outputs 0^{n^2}.

Transitions that are not shown are assumed to go to the reject state, as discussed in class. However, these never occur. We have ommitted the reject state as it is never reached.

$-$ in the transitions represents any character. The transitions $\neg x/-, L$ or $\neg x/-, R$ denote that upon reading any character $y \neq x$, replace it with y itself and move left or right, respectively (as in the TM given in problem 5).

The machine works as follows:

- Mark the first 0 as 'final' by changing it to 1.
- For each remaining 0, append two 1s to the right. When this is done, change another one of the 0s to 1.
- When no more 0s are left, change all 1s to 0s and accept.

This uses the fact that

$$n^2 = n + \sum_{i=1}^{n-1} 2i.$$

■

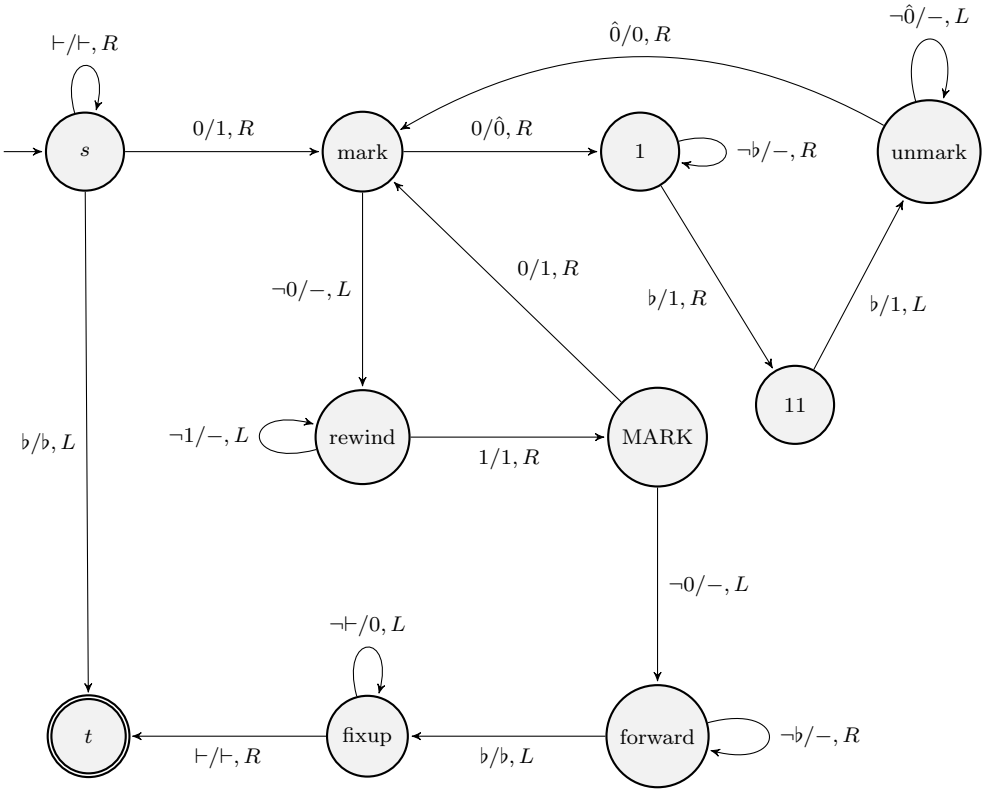

Figure 1: A Turing machine that computes the square function.

Problem 5.2. 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. 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, which 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 5.3. Let $L, K \subseteq \Sigma^*$. Define

$$L/K = \{x \mid \exists y \in K, xy \in L\}.$$

(a) Show that if L is regular and K is *any* language, then L/K is regular.

(b) Show that even if we are given a DFA for L and a Turing machine for K, we cannot always construct an automaton for L/K.

Solution. Let L be regular. Its canonical relation $\equiv_L$ has finite index.

Consider the relation $\equiv_{L/K}$ defined by

$$x \equiv_{L/K} y \iff (\forall z \in \Sigma^*, xz \in L/K \iff yz \in L/K.)$$

We will show that

$$x \equiv_L y \implies x \equiv_{L/K} y.$$

If this is true, then $\equiv_L$ is a refinement of $\equiv_{L/K}$, so $\equiv_{L/K}$ has finite index.

Let $x \equiv_L y$. Then for all $z \in \Sigma^*$, $xz \in L \iff yz \in L$. Fix a $z \in \Sigma^*$. Suppose $xz \in L/K$. Then there exists $w \in K$ such that $xzw \in L$. But then $yzw \in L$ (by the canonical relation), so $yz \in L/K$. Symmetrically, if $yz \in L/K$, then $xz \in L/K$. Thus for all $z \in \Sigma^*$, $xz \in L/K \iff yz \in L/K$, so that $x \equiv_{L/K} y$. This proves the claim. ■

Problem 5.4. Show that neither the language

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

nor its complement is recursively enumerable.

Solution. Both proofs are by reducing $\neg\mathsf{HP}$ to the given language. The reduction for $\neg$TOTAL is easier, so we present that proof first.

$\neg\mathsf{HP} \leq \neg$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

$$M\#x \in \neg\mathsf{HP} \iff \varphi(M\#x) \in \neg\text{TOTAL}.$$

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

$$M\#x \in \neg\mathsf{HP} \iff Q_{M,x} \in \text{TOTAL}.$$

This proves the claim. □

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

Problem 5.5. Consider the TM M given in the assignment, with input alphabet $\{a, b\}$.

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

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

(c) Justify the claim that for two valid configurations c_1 and c_2 of M, which are of the same length, we have $c_1 \overset{1}{\Rightarrow} c_2$ iff for each position in c_1, the triple of symbols in c_1 and the corresponding triple in c_2 match.

(d) Define an analagous notion of matching pairs of symbols. What if we weaken the criterion above to say that at each position the pairs of symbols in c_1 and c_2 "match"?

Solution.

(a)

$\vdash$	b	a	a	b	b	$\flat$	$\#$	$\vdash$	b	a	a	b	b	$\flat$	$\#$	$\vdash$	b	a	a	b	b	$\flat$	$\#$
s	$-$	$-$	$-$	$-$	$-$	$-$	$-$	s	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	p	$-$	$-$	$-$	$-$	$-$	$-$
$\vdash$	b	a	a	b	b	$\flat$	$\#$	$\vdash$	b	a	a	b	b	$\flat$	$\#$	$\vdash$	b	a	a	b	b	$\flat$	$\#$
$-$	$-$	$-$	p	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	p	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	p	$-$	$-$
$\vdash$	b	a	a	b	b	$\flat$	$\#$	$\vdash$	b	a	a	b	b	c	$\#$	$\vdash$	b	a	a	b	c	c	$\#$
$-$	$-$	$-$	$-$	$-$	$-$	p	$-$	$-$	$-$	$-$	$-$	$-$	q	$-$	$-$	$-$	$-$	$-$	$-$	t	$-$	$-$	$-$

(b) We assume the unspecified transitions to be $-/-, R$ into the reject state r (as discussed on MS Teams). 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 $\langle$triple 1 | triple 2$\rangle$ for clarity.

The matching triples for s are

$$\left\langle \begin{array}{ccc|ccc} \vdash & x & y & \vdash & x & y \\ s & - & - & - & s & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & \vdash & y & x & \vdash & y \\ - & s & - & - & - & s \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & y & \vdash & x & y & \vdash \\ - & - & s & - & - & - \end{array} \right\rangle$$

$$\left\langle \begin{array}{ccc|ccc} a & x & y & a & x & y \\ s & - & - & - & - & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & a & y & x & a & y \\ - & s & - & r & - & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & y & a & x & y & a \\ - & - & s & - & r & - \end{array} \right\rangle$$

$$\left\langle \begin{array}{ccc|ccc} b & x & y & b & x & y \\ s & - & - & - & p & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & b & y & x & b & y \\ - & s & - & - & - & p \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & y & b & x & y & b \\ - & - & s & - & - & - \end{array} \right\rangle$$

$$\left\langle \begin{array}{ccc|ccc} c & x & y & c & x & y \\ s & - & - & - & r & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & c & y & x & c & y \\ - & s & - & - & - & r \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & y & c & x & y & c \\ - & - & s & - & - & - \end{array} \right\rangle$$

$$\left\langle \begin{array}{ccc|ccc} \flat & x & y & \flat & x & y \\ s & - & - & - & r & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & \flat & y & x & \flat & y \\ - & s & - & - & - & r \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & y & \flat & x & y & \flat \\ - & - & s & - & - & - \end{array} \right\rangle$$

For p, the matching triples are

$$\left\langle \begin{array}{ccc|ccc} \flat & x & y & c & x & y \\ p & - & - & - & - & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & \flat & y & x & c & y \\ - & p & - & q & - & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & y & \flat & x & y & c \\ - & - & p & - & q & - \end{array} \right\rangle$$

$$\left\langle \begin{array}{ccc|ccc} \beta & x & y & \beta & x & y \\ p & - & - & - & p & - \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & \beta & y & x & \beta & y \\ - & p & - & - & - & p \end{array} \right\rangle \quad \left\langle \begin{array}{ccc|ccc} x & y & \beta & x & y & \beta \\ - & - & p & - & - & - \end{array} \right\rangle$$

For q, they are

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

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

For r, they are

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

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

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

In addition to these, we have the matching triples

$$\left\langle \begin{array}{ccc|ccc} x & y & z & x & y & z \\ - & - & - & \theta & - & - \end{array} \right\rangle \qquad \left\langle \begin{array}{ccc|ccc} x & y & z & x & y & z \\ - & - & - & - & - & \theta \end{array} \right\rangle$$

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

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

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

(c) The construction of the triples makes it clear that if $c_1 \overset{1}{\Rightarrow} c_2$, 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 triple with no state symbols is

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

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

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

The match

$$\left\langle \begin{array}{ccc|ccc} u & v & w & u & v & w \\ - & - & q & \prime & - & - \end{array} \right\rangle$$

ensures that the letter change and move are consistent. But what forces the matching triple

$$\left\langle \begin{array}{ccc|ccc} x & y & z & x & y & z \\ - & - & - & q & - & - \end{array} \right\rangle?$$

The state could 'vanish' in the false transition

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

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

$$\left\langle \begin{array}{ccc|ccc} v & w & x & v & w & \prime \\ - & q & - & x & - & - \end{array} \right\rangle$$

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

(d) The desirable properties still hold. If c_1 and c_2 are valid configurations of the same length, then $c_1 \overset{1}{\Rightarrow} c_2$ 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

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

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 5.6. Show that it is undecidable whether the intersection of two given CFLs is a CFL.

Solution. We will reduce the halting problem to this problem.

Let M be a 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,

- M_1 checks that the even-numbered configurations are correctly followed by their successors.
- M_2 checks that the odd-numbered configurations are correctly followed by their successors.

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

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