Indian Institute of Technology Palakkad
Department of Computer Science and Engineering
CS5616 Computational Complexity
January May 2024

IIT PALAKKAD

Problem Set – 1 | Total Points – 50
Name: Neeraj Krishna N | Given on 03 Feb
Roll no: 112101033 | Due on 11 Feb

Instructions

- Use of resources other than class notes and references is forbidden.
- Collaboration is not allowed. Credit will be given for attempts and partial answers.

1. (15 points) [**More properties of R and RE**] We saw in class that recursive languages are closed under complement. We will see more properties below.
   (a) (5 points) Show that recursively enumerable languages are closed under union and intersection.
   (b) (5 points) For a language L, and $i \geq 1$, $L^i = \{a_1 a_2 \ldots a_i \mid a_1, \ldots, a_i \in L\}$. The Kleene closure of L, denoted by L^*, is defined as $\{\epsilon\} \cup \bigcup_{i \geq 1} L^i$. Show that recursive and recursively enumerable languages are closed under Kleene closure.
   (c) (5 points) For languages $L_1, L_2 \subseteq \Sigma^*$, define $L_1/L_2 = \{x \in \Sigma^* \mid \exists y \in L_2 \; xy \in L_1\}$. Show that if L_1 and L_2 are recursively enumerable, then so is L_1/L_2.

Solution:

(a) 1. **To show** : recursively enumerable are closed under union

Let L_1 and L_2 be the 2 recursively enumerable languages and let the turing machines (not necessarily total) accepting them be M_1 and M_2 respectively. Now we need to give a turing machine M, which accepts $L = L_1 \cup L_2$.
The description of M is as follows:
" On input x,
(a) Run one step of M_1 on x
(b) Run one step of M_2 on x
(c) As long as either of them does not accept, repeat steps (a) and (b)
(d) If one of them accept, accept and halt
"

To prove : M accepts $L = L_1 \cup L_2$

(a) *Proof of* $L \subseteq L(M)$:
$x \in L$
$\implies x \in L_1 \cup L_2$
$\implies x \in L(M_1) \vee x \in L(M_2)$ (since $L_1 = L(M_1)$ and $L_2 = L(M_2)$)
$\implies$ Either M_1 or M_2 will halt and accept (and since we are performing

one step in each machine, we will not be stuck in a loop, even if one of the machine loops on the given input)
$\implies M$ halts and accepts
$\implies x \in L(M)$

(b) *Proof of* $L(M) \subseteq L$:
$x \in L(M)$
$\implies M$ accepts x
$\implies M_1$ accepted x and M reached (d) in the algorithm
$\vee\ M_2$ accepted x and M reached (d) in the algorithm
$\implies x \in L(M_1) \vee x \in L(M_2)$
$\implies x \in L_1 \vee x \in L_2$ (since $L_1 = L(M_1)$ and $L_2 = L(M_2)$)
$\implies x \in L_1 \cup L_2$
$\implies x \in L$

Hence Proved

2. **To show** : recursively enumerable are closed under intersection

Let L_1 and L_2 be the 2 recursively enumerable languages and let the turing machines (not necessarily total) accepting them be M_1 and M_2 respectively.
Now we need to give a turing machine M, which accepts $L = L_1 \cap L_2$.
The description of M is as follows:
" On input x,

(a) Run one step of M_1 on x
(b) Run one step of M_2 on x
(c) As long as both of them do not halt, repeat step (a) and step (b)
(d) If both of them accept, accept and halt

"

To prove : M accepts $L = L_1 \cap L_2$

(a) *Proof of* $L \subseteq L(M)$:
$x \in L$
$\implies x \in L_1 \cap L_2$
$\implies x \in L(M_1) \wedge x \in L(M_2)$ (since $L_1 = L(M_1)$ and $L_2 = L(M_2)$)
$\implies$ By the description of the machine, since both M_1 and M_2 accepts, M accepts (and since we are performing one step in each machine, we will not be stuck in a loop, even if one of the machine loops on the given input)
$\implies M$ halts and accepts
$\implies x \in L(M)$

(b) *Proof of* $L(M) \subseteq L$:
$x \in L(M)$
$\implies M$ accepts x
$\implies M_1$ accepted $x \wedge M_2$ accepted x
and hence M reached (d) in the algorithm
$\implies x \in L(M_1) \wedge x \in L(M_2)$
$\implies x \in L_1 \wedge x \in L_2$ (since $L_1 = L(M_1)$ and $L_2 = L(M_2)$)
$\implies x \in L_1 \cap L_2$
$\implies x \in L$

Hence Proved

(b) Let M be the turing machine which accepts L

Let M_k denote a k-tape Turing machine whose description is as follows,

$M_k =$

"

On input $\langle x_1, x_2, \ldots, x_k \rangle$,

(a) In a time shared fashion, On tape T_i where $i \in \{1, \ldots, k\}$, Run M on x_i

(b) Accept if all machines accept

"

Let $L_k = \{\langle x_1, x_2, \ldots, x_k \rangle \mid x_i \in L \; \forall i \in \{1, \ldots, k\}\}$

Proof of $L(M_k) = L_K$

$\langle x_1, x_2, \ldots, x_k \rangle \in L(M_k)$

$\Leftrightarrow x_i \in L \forall i \in \{1, \ldots, k\}$

$\Leftrightarrow$ The machine on tape T_i accepts for all i

$\Leftrightarrow M_k$ accepts

$\Leftrightarrow \langle x_1, x_2, \ldots, x_k \rangle \in L(M_k)$

Let us describe a turing machine M^* with infinite tapes, which accepts the language L^*

$M^* =$

"

On input x,

(a) If $x = \epsilon$, accept

(b) Else $x = x_1 x_2 x_3 \ldots x_n$ where $n = |x|$

  (a) Let the tapes be denoted by $T_{(i,k)}$ where k denotes the number of splits of string x and $i \in \binom{n-1}{k}$ denotes the different possible splits
  (For example, if $k = 1$, then $T_{(i,1)}$ denotes the string $\langle x_1 x_2 \ldots x_i, x_{i+1} \ldots x_n \rangle$)

  (b) For all $k \in \{0 \ldots, n-1\}$ and for all $i \in \{1, \ldots, \binom{n-1}{k}\}$ Run M_{k+1} on the tape $T_{(i,k)}$ in a time shared fashion

  (c) Accept if any one of the machines accept

"

Proof of $L(M^*) = L^*$

$x \in L^*$

$\Rightarrow$There exists a way of splitting the string x using k splits, for some k for which $\langle x_1, x_2, \ldots, x_{k+1} \rangle \in L_{k+1}$ (or $x = \epsilon$ in which case the machine accepts)

$\Rightarrow$Since the Turing machine is checking all possible splits, it is bound to find some (k, i) such that the machine M_{k+1} running on tape $T_{(i,k)}$ accepts

$\Rightarrow M^*$ accepts x

$\Rightarrow x \in L(M^*)$

$x \in L(M^*)$

$\Rightarrow M^*$ accepted x

$\Rightarrow$There existed a (k, i) such that the machine M_{k+1} running on tape $T_{(i,k)}$ accepted

$\Rightarrow$That (i, k) is a valid split of the string $x = x_1 x_2 \ldots x_{k+1}$

such that $\langle x_1, x_2, \ldots, x_{k+1} \rangle \in L_{k+1}$

$\Rightarrow x \in L^*$

Hence $L(M^*) = L^*$

We already know that infinite tape turing machine are equivalent to single tape turing machines, Hence the above construction of M^* is a valid turing machine construction

Since, nowhere in the above proof did we assume that the turing machine is total, Hence the same reduction works for both recursive and recursively enumerable languages

The only difference being, if it is recursive, all the machines described above **will be** total, since every step halts at some Points

Hence proved that recursive and recursively enumerable languages are closed under Kleene closure

(c) Let the Turing machines (not necessarily total) that accepts L_1 and L_2 be M_1 and M_2

Let $L = L_1/L_2 = \{x \in \Sigma^* \mid \exists y \in L_2,\ xy \in L_1\}$

Let us first define another Turing Machine M'' which takes the input $\langle x, y \rangle$ M'' =

"

On input $\langle x, y \rangle$,

(a) Runs M_1 on xy and M_2 on y in a time shared fashion

(b) If both M_1 and M_2 accept, accept

"

Let $L'' = \{\langle x, y \rangle \mid xy \in L_1, y \in L_2\}$

We will prove that $L(M'') = L''$

$\langle x, y \rangle \in L(M'')$

$\Rightarrow M''$ accepted $\langle x, y \rangle$

$\Rightarrow M_1$ accepted xy and M_2 accepted y

$\Rightarrow xy \in L(M_1)$ and $y \in L(M_2)$

$\Rightarrow xy \in L_1$ and $y \in L_2$ since $L(M_1) = L_1$ and $L(M_2) = L_2$

$\Rightarrow \langle x, y \rangle \in L''$

$\langle x, y\rangle \in L''$

$\Rightarrow xy \in L_1$ and $y \in L_2$

$\Rightarrow xy \in L(M_1)$ and $y \in L(M_2)$ since $L(M_1) = L_1$ and $L(M_2) = L_2$

$\Rightarrow M_1$ accepts xy and M_2 accepts y

$\Rightarrow M''$ accepts $\langle x, y\rangle$

$\Rightarrow \langle x, y\rangle \in L(M'')$

Now we will construct an infinite tape turing machine M_∞ which accepts L (Since we know that infinite tape turing machine can be simulated using a single tape turing machine, we can prove that L is recursive enumerable)

Let us index the tapes as T_ϵ, T_0, T_1, T_{00}, T_{01}, T_{10}, T_{11}, ...

$M_\infty =$

"

On input x,

(a) In each tape T_y, Run the machine M'' on the input $\langle x, y\rangle$ $\forall y \in \Sigma^*$ in a time shared fashion

(b) If any of the machine accepts, accept

"

Let us prove that $L(M_\infty) = L$

Proof of $L(M_\infty) \subseteq L$

$x \in L(M_\infty)$

$\Rightarrow M_\infty$ accepts x

$\Rightarrow$There existed a $y \in \Sigma^*$ for which the machine M'' (which was running on the tape T_y) accepted the input $\langle x, y\rangle$

$\Rightarrow \langle x, y\rangle \in L''$

$\Rightarrow x \in L$

Proof of $L \subseteq L(M_\infty)$

$x \in L$

$\Rightarrow$There exists a $y \in \Sigma^*$ for which $\langle x, y\rangle \in L''$

$\Rightarrow$In the machine M_∞, there exists a tape T_y where the turing machine M'' running on that tape will accept

$\Rightarrow M_\infty$ will accept x

$\Rightarrow x \in L(M_\infty)$

2. (10 points) [**Non-determinism**] The machines that we saw in class have a single valued transition function and hence are *deterministic*. A nondeterministic Turing machine is a machine where the transition function can take multiple values.

   (a) (5 points) Give a rigorous formal definition of a non-deterministic Turing machine, including a definition of configuration, next configuration relation and acceptance.

(b) (5 points) Argue that deterministic machines and non-deterministic machines are equivalent (in the sense that they can simulate each other).

Solution:

(a) Non-deterministic Turing Machine is a 9-tuple $N = (Q, \Sigma, \Gamma, \vdash, \sqcup, \Delta, s, t, r)$ where

- Q is a finite set of states
- Σ is a finite set of input alphabets
- Γ is the finite set of tape symbols
- $\vdash \in \Gamma \setminus \Sigma$ is the left end marker
- $\sqcup \in \Gamma \setminus \Sigma$ is the blank symbol
- $\Delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$ is the transition relation
- $s \in Q$ is the start state
- $t \in Q$ is the accept state
- $r \in Q$ is the reject state

Configuration :
A configuration c is a tuple of $(q, w\sqcup^\omega, n)$ where $q \in Q$, $w \in \Gamma^*$, $n \in \mathbb{N}$

Next Configuration Relation :
It is the single step transition from one configuration to a set of configurations. $(p, z, n) \to (q, z', n')$ where $n' = n+1$ if head moved right, else if head moved left, $n' = n - 1$, else $n' = n$ and $((p, z), (q, z')) \in \Delta$

Acceptance :
Turing machine M, is said to accept an input x, if after a finite number of steps, the configuration becomes (t, y, n) for some $y \in \Gamma^*$ and $n \in \mathbb{N}$

(b) 1. Let $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ be a deterministic Turing Machine
We want to construct an equivalent Turing Machine
$N = (Q', \Sigma', \Gamma', \vdash, \sqcup, \Delta, s', t', r')$
Now let
$Q' = Q$
$\Sigma' = \Sigma$
$\Gamma' = \Gamma$
$\Delta(p, z) = \{\delta(p, z)\}$
$s' = s$
$t' = t$
$r' = r$
Now we have to prove $L(M) = L(N)$
Proof of $L(M) \subseteq L(N)$:
$x \in L(M)$
$\Rightarrow$There is set of steps for M which leads to terminal accept state t
$\Rightarrow$Since Δ has only one transition for every possible tuple (p, z), N will also take the same steps as M

$\Rightarrow x \in L(N)$

Proof of $L(N) \subseteq L(M)$:

$x \in L(N)$

$\Rightarrow$There is a set of steps for which N reaches t'

$\Rightarrow$Since Δ has only one transition for every possible tuple (p, z), it behaves like a deterministic turing machine

$\Rightarrow M$ also takes the exact same transitions for input x and reaches the accept state t

$\Rightarrow x \in L(M)$

Hence $L(M) = L(N)$ and we have proved that every deterministic Turing Machine can be simulated using a Non deterministic Turing Machine

2. Let $N = (Q, \Sigma, \Gamma, \vdash, \sqcup, \Delta, s, t, r)$ be a non deterministic Turing Machine

   We want to construct an equivalent deterministic Turing Machine

   $M = (Q', \Sigma', \Gamma', \vdash, \sqcup, \delta, s', t', r')$ such that it accepts the same language as that of N

   Let

   $Q' = 2^Q$

   $\Sigma' = \Sigma$

   $\Gamma' = \Gamma$

   $s' = \{s\} \in Q'$

   $t' = \{t\} \in Q'$

   $r' = \{r\} \in Q'$

   Now for δ,

   Let $q' = \{q_1, q_2, \ldots, q_i\}$ for some $i \in \{0, 1, \ldots, |Q|\}$, where each $q_i \in Q$ (if $i = 0$, then $q' = \phi$)

   $\delta(q', z) = \bigcup_{q_i \in Q} \Delta(q_i, z)$

   Now, we have to prove $L(N) = L(M)$

   Proof of $L(N) \subseteq L(M)$:

   $x \in L(N)$

   $\Rightarrow$There exists a non deterministic set of choices for the machine N which leads to the terminal state t'

   $\Rightarrow$Since, M is in essence, simulating every possible transitions of N, there exists a transition $\delta(\cdot)$ for which $(t, z) \in \delta(\cdot)$ for some z

   $\Rightarrow M$ accepts x

   $\Rightarrow x \in L(M)$

   $x \in L(M)$

   $\Rightarrow$There exists a set of transitions at the end of which $(t, z) \in \delta(\cdot)$

   $\Rightarrow$Then, if we take the same set of non deterministic choices in the non deterministic turing machine, we will reach t in N

   $\Rightarrow x \in L(N)$

   Hence showed that $L(N) = L(M)$ and hence every non deterministic turing machine can be simulated by a deterministic turing machine

   (This construction works only because all of the following are finite : Q, Σ, Γ)

3. (15 points) [**Undecidability**]

(a) (5 points) Consider the problem of deciding if two given Turing machines accept the same set. Formulate this as a language and show that it is undecidable.

(b) (5 points) Consider the problem of deciding given a Turing machine and a state, whether it enters the state on some input. Formulate this as a language and show that it is undecidable.

(c) (5 points) Show that if Membership problem is undecidable, then Halting problem is undecidable. [*Note: We independently know the undecidability of both the problems. Nevertheless, this question asks to prove the implication.*]

Solution:

(a) Let the language be

$L = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$

We will show that if L is decidable, then $HP^{\complement}$ is decidable

Let M_σ be a machine such that on input $\langle M, x\rangle$, it produces a description of 2 machines $\langle M_1, M_2\rangle$ which are as follows

Description of M_1 :

" On input y,

(a) Run M on x

(b) If M halts on x, Accept

"

Description of M_2 :

" On input y,

(a) Run M on x

(b) If M halts on x, Reject

"

We will prove that $\langle M, x\rangle \in HP \Leftrightarrow \langle M_1, M_2\rangle \notin L$

Proof of forward direction

$\langle M, x\rangle \in HP$

$\Rightarrow M$ halts on x

$\Rightarrow M_1$ accepts all strings and M_2 rejects all strings

$\Rightarrow L(M_1) = \Sigma^*$ and $L(M_2) = \phi$

$\Rightarrow L(M_1) \neq L(M_2)$

$\Rightarrow \langle M_1, M_2\rangle \notin L$

Proof of reverse direction

$\langle M, x\rangle \notin HP$

$\Rightarrow M$ does not halt on x

$\Rightarrow L(M_1) = \phi$ and $L(M_2) = \phi$

$\Rightarrow L(M_1) = L(M_2)$

$\Rightarrow \langle M_1, M_2 \rangle \in L$

Hence, L is undecidable

(b) Let the language be $L = \{\langle M, q \rangle \mid M \text{ enters state } q \text{ on some input}\}$

We will show that if L is decidable, then HP is decidable

Let M_σ be a machine such which produces an instance of L, given an instance of HP , i.e on input $\langle M, x \rangle$, M_σ outputs $\langle M', t \rangle$ where t is the accept state of M' and the description of M' is as follows :

"

On input y,

(a) Run M on x

(b) If M halts on x, then enter accept state and accept

"

We will now show that $\langle M, x \rangle \in HP \Leftrightarrow \langle M', t \rangle \in L$

Proof of forward direction :

$\langle M, x \rangle \in HP$

$\Rightarrow$ M halts on x

$\Rightarrow$ M' enters t and accepts for all inputs y

$\Rightarrow$ $\langle M, t \rangle \in L$

Proof of reverse direction :

$\langle M, x \rangle \notin HP$

$\Rightarrow$ M does not halt on x

$\Rightarrow$ M' loops and never enters state t for any input

$\Rightarrow$ $\langle M', t \rangle \notin L$

Hence L is undecidable

(c) We need to show that if MP is undecidable, then HP is undecidable which is same as proving its contrapositive, i.e

HP is decidable $\Rightarrow MP$ is decidable

Let M_σ be a machine which on input $\langle M, x \rangle$ (an instance of MP) outputs $\langle M', \epsilon \rangle$ (which is an instance of HP) where the description of M' is as follows:

"

On input y,

(a) Run M on x

(b) If M accepts x, halt

(c) If M rejects x, loop

”

Now, we will show that $\langle M, x\rangle \in MP \Leftrightarrow \langle M', \epsilon\rangle \in HP$

Proof of forward direction :

$\langle M, x\rangle \in MP$

$\Rightarrow M$ accepts x

$\Rightarrow M'$ halts on every input, in particular ϵ also

$\Rightarrow \langle M', \epsilon\rangle \in HP$

Proof of reverse direction :

$\langle M, x\rangle \notin MP$

$\Rightarrow M$ does not accept x

$\Rightarrow M$ rejects x or M loops on x

$\Rightarrow$If M rejects x, then M' loops on every input (in particular ϵ) and

If M loops on x, then also M' gets stuck in step (a) and M' loops on every input (in particular ϵ)

$\Rightarrow \langle M', \epsilon\rangle \notin HP$

Hence proved that if MP is undecidable, then HP is undecidable

4. (10 points) [**Decidable or Undecidable ?**] Let $|M|$ denote the length of the Turing machine description. Are the following problems decidable ? Justify.
   (a) (5 points) Does a Turing machine M takes at least $|M|$ steps on *some* input ?
   (b) (5 points) Does a Turing machine M takes at least $|M|$ steps on *all* inputs ?

Solution:

(a) This is decidable because every Turing Machine takes atleast $|M|$ steps on some inputs. Below is the justification

Let us take a Turing Machine M_σ whose description is as follows

“

On input M,

1. If M is not a valid encoding of Turing Machine, reject
2. Else accept

”

The above machine accepts any valid encoding of Turing Machine. The above machine is total because there exists a **total turing machine** which checks if a string is a valid encoding of a Turing Machine

Let us take an arbitrary Turing Machine M
Let $n = |M|$
Let string x be $0^{|M|+1}$
For this string, the Turing Machine will take atleast $|M|$ steps
Hence for every Turing Machine M, there exists an input for which M will take atleast $|M|$ steps.

(b) Let the language be $L = \{M \mid M \text{ takes at least } |M| \text{ steps on all inputs}\}$

This is undecidable because, we can show that if the given problem is decidable, then HP is decidable. Let us take a turing machine M_σ which on input $\langle M, x\rangle$, provides a description of another turing machine M' which is
" On input y,

(1) Run M on x

(2) If M halts on x, perform $|M'|$ steps and accept

"

We will prove that $\langle M, x\rangle \in HP \Leftrightarrow M' \in L$

Proof of forward direction
$\langle M, x\rangle \in HP$
$\Rightarrow M$ halts on x
$\Rightarrow M'$ reaches step (2) and performs atleast $|M'|$ steps and then accepts
$\Rightarrow M' \in L$

Proof of reverse direction
$\langle M, x\rangle \notin HP$
$\Rightarrow M$ does not halt on x
$\Rightarrow M'$ is stuck in step 1 itself
$\Rightarrow M'$ does not perform even a single step on any input
$\Rightarrow M' \notin L$

Since HP is undecidable, L is undecidable