INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD
Department of Computer Science and Engineering
CS5616 Computational Complexity
January May 2024

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 $i \geq 1$, $L^i = \{a_1 a_2 \dots a_i \mid a_1, \dots, a_i \in L\}$. 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: Write your answer here.

(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 1a and 1b
(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 1d in the algorithm
$\vee$ M_2 accepted x and M reached 1d 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 2a and step 2b
(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 2d 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)

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:

1. Non-deterministic Turing Machine is a 9-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ 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 $n \in \mathbb{N}$

2.

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:

1.
2.
3.

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:

1. 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,

   (a) If M is not a valid encoding of Turing Machine, reject

   (b) 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.

2. ??????
   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 " "