# COL352
# Problem Sheet 2

Himanshi Ghai (2019CS50433)
Mallika Prabhakar (2019CS50440)
Sayam Sethi (2019CS10399)

February 2022

## Contents

## 1 Question 1

Question 1

**Question.** *Show that every infinite Turing-recognizable language has an infinite decidable subset.*

*Proof.* Let the Turing-recognizable language be $L$. Consider an enumerator for this language $E$. Consider the following algorithm:

**Algorithm 1** Return the first word of length $n$ enumerated such that all words enumerated earlier are of shorter length else $\epsilon$

```
procedure EnumerateFirst($E, n$)
    $w \leftarrow \epsilon$
    $curr_w \leftarrow \epsilon$
    while $|curr_w| < n$ do
        $curr_w \leftarrow next(E)$
        if $|curr_w| > |w|$ then
            $w \leftarrow curr_w$
        end if
    end while
    if $|w| > n$ then
        $w \leftarrow \epsilon$
    end if
    return $w$
end procedure
```

We now define the following Turing machine $T_{decidable}$:

**Algorithm 2** Turing machine of a decidable subset

```
procedure DecidableSubset($L, w$)
    $E \leftarrow enumerator(L)$
    $w' \leftarrow EnumerateFirst(E, |w|)$
    if $w' = w$ then
        return accept
    else
        return reject
    end if
end procedure
```

We claim that $T_{decidable}$ is a Turing machine and decidable. For this, we first prove that Algorithm 1 is correct and terminates. For correctness, notice that the while loop exits only if $|curr_w| \geq n$ and thus the length of the string in $w$ after the termination of the while loop is at least $n$. Therefore, the final returned value is the first string enumerated with length equal to $n$ (such that all strings before are of smaller length) or $\epsilon$.
For termination, notice that there are only a finite possibilities before $E$ enumerates a string with length $\geq n$. Therefore, the while loop terminates after a finite number of iterations.

We have shown that Algorithm 1 is decidable since it halts. Therefore, the string $w'$ in Algorithm 2 is generated in finite number of steps. The comparison of $w'$ and $w$ also completes in finite steps. Therefore, Algorithm 2 accepts or rejects $w$ in finite number of steps. Therefore, $T_{decidable}$ is a decidable Turing machine.
We also need to show that $T_{decidable}$ recognizes an infinite language. This is obvious since $L$ is infinite and therefore the following set forms an infinite sequence:

$$\{i | \forall i \in \mathbb{N} : EnumerateFirst(E, i) \neq \epsilon\} \tag{1}$$

This set is a bijection to the words recognised by $T_{decidable}$. Therefore, $L(T_{decidable})$ is also an infinite set.
Thus, we have shown that for every infinite Turing-recognizable language, we have an infinite decidable subset. □

## 2 Question 2

Question 2

**Question.** *Show that single-tape TMs that cannot write on the portion of the tape containing the input string recognize only regular languages.*

*Proof.* We consider the subset $L'$ of $L(M)$ such that $w \in L'$ leaves the input portion at least once. The language $L(M) \setminus L'$, can be simulated by a 2DFA and therefore this language is regular. Therefore, it is enough to show that the language $L'$ is regular (since union of regular languages is regular). Define the following (partial) function $f_w$ for such a Turing machine $M$ and input $w$:

$$\begin{aligned} & f_w(q) = q', \text{ when } M \text{ enters the input portion on } q \text{ and exits the input portion on } q' \\ & f_w(q) = q_{accept}, \text{if } M \text{enters input portion on } q, \text{ never exits and accepts } w \\ & f_w(q) = q_{reject}, \text{if } M \text{enters input portion on } q, \text{ never exits and rejects } w \end{aligned} \tag{2}$$

It is easy to see that acceptance is decidable if $M$ doesn't exit the input portion. This is because there are only a finite number of states $|Q|$ and a finite number of positions $|w|$. Therefore, if $w$ isn't accepted within $|Q|^{|w|}$ steps after entering the input portion, we can reject $w$.
We consider the following equivalence relation:

$$[x] \equiv [y] \iff f_x = f_y \tag{3}$$

It is easy to see that this equivalence relation has a finite number of equivalence classes since total number of functions are $|Q|^{|Q|+2}$. We will now show that this relation also satisfies the property of right congruence:

$$\begin{aligned} & [x] \equiv [y] \\ \implies & \text{every time } M \text{ goes from position } |w| \text{ to } |w|+1, \text{ it is at the same state for both } x, y \\ \implies & [xa] \equiv [ya] \end{aligned} \tag{4}$$

Therefore, $f_w$ defines an equivalence, right congruent relation with finite classes. Therefore, by Myhil-Nehrode theorem, $L'$ recongizes only regular languages. From the closure of regular languages under union, we get that $L(M)$ and therefore $M$, recognizes only regular languages. Hence, proved. □

## 3 Question 3

Question 3

**Question.** *Let $C$ be a language. Prove that $C$ is Turing-recognizable iff a decidable language $D$ exists such that*

$$C = \{x|\exists y(\langle x, y\rangle \in D)\}$$

*Proof.* To prove the iff statement, we consider the following parts one by one:

*Direction 1:* Given a decidable language $D$, the Language $C = \{x|\exists y(\langle x, y\rangle \in D)\}$ formed is Turing recognizable.
**Proof:** For a language to be Turing recognizable, if a string $x \in \Sigma^*$ the TM halts and accepts it and if $x \notin \Sigma^*$, TM either rejects it or keeps running.
In our case, Language $D$ is Turing decidable, which means given an input $\langle x, y\rangle$ it either accepts or rejects the string and halts for every input.
Given any string $x$, $C$ accepts it if there exists a string $y$ such that $\langle x, y\rangle \in D$. Hence our Turing Machine for $C$ keeps searching for a string $y$ given an input $x$ so that $\langle x, y\rangle \in D$. Since $y$ is of finite length it can always be found in finite amount of time, if such $y$ is found, $x$ is an accepted string and the TM halts. If no such $y$ is found, TM keeps looking for it indefinitely.
This implies that language $C$ is Turing recognizable □

*Direction 2:* Given language $C$ such that $C$ is Turing recognizable, then there exists a Turing-decidable language $D$ such that $C = \{x|\exists y(\langle x, y\rangle \in D)\}$.
**Proof:** We define the following algorithm for the language $D$:

**Algorithm 3** Algorithm for a language $D$

```
1: procedure D($C, x, y$)
2:     $status \leftarrow run_n(C, x, |y|)$        ▷ $run_n(M, w, n)$ simulates $M$ with input $w$ for atmost $n$ steps
3:     if $status = accepted$ then
4:         return accept
5:     else
6:         return reject
7:     end if
8: end procedure
```

Since Algorithm 3 simulates $C$ on $x$ for *atmost* $|y|$ iterations, the algorithm always terminates in finite number of steps. Also, it uniquely accepts or rejects $\langle x, y\rangle$. Therefore, the language $D$ is decidable. Also, it is easy to see that the language satisfies the property that $C$ accepts $x$ iff $\exists y : \langle x, y\rangle \in D$. Therefore, we have shown the existence of such a language. □

On combining both proven statements in Direction 1 and Direction 2, we can say that given language $C = \{x|\exists y(\langle x, y\rangle \in D)\}$ is Turing recognizable if language $D$ is Turing Decidable.

*Hence Proved*

□

## 4 Question 4

Question 4

**Question.** *Say that a variable $A$ in CFL $G$ is is* usable *is it appears in some derivation of some string $\omega \in G$. Given a CFG $G$ and a variable $A$, consider a problem of testing whether $A$ is usable. Formulate this problem as language and show that is decidable.*

*Solution.* We need to test here whether the start symbol $S$ of grammar $G$ generates a string whose derivation contains variable $A$. In order to do so we design an algorithm to test the above. The algorithm will first mark $\dot{}$ on all the symbols that generate strings (Ex: $X$ as $\dot{X}$). Then starting from symbol $A$, it will underline the variables that will generate string(marked with dot) and also contains $A$ in at least one of the derivation (Ex: $\dot{X}$ as $\underline{\dot{X}}$). Now, if the start symbol $S$ is marked as $\underline{\dot{S}}$, it means that start symbol generate a string that contains $A$ in its derivation.

We use this algorithm to design a Turing machine $R$ which decides the following language $U_{TM}$.

$$U_{TM} = \{\langle G, A\rangle | A \text{ is a variable which is } \textit{usable} \text{ in } G\}$$

Turing machine $R$ works as following:

$S =$ "On input $\langle G, A\rangle$, where $G$ is a grammar and $A$ a variable in $G$:

1. Mark all terminal symbol in $G$.
2. Repeat until no new symbol is marked with "$\dot{}$":
   (a) Mark all variable $X$ where $G$ has a rule $X \longrightarrow U_1U_2U_3...U_k$ and each symbol $U_i$ has been marked as $\dot{U}_i$.
3. If $A$ not marked as $\dot{A}$ reject else mark $\dot{A}$ as $\underline{\dot{A}}$.
4. Repeat until no new symbol is underlined
   (a) Underline all variable $\dot{X}$ where $G$ has a rule $X \longrightarrow U_1U_2U_3...U_k$ and each symbol $U_i$ has been marked as $\dot{U}_i$ and at least one of symbol $U_j$ has been underlined.
5. If start symbol of $G$ is marked as $\underline{\dot{S}}$ then accept; else reject.

In first loop, marking variables with "$\dot{}$" tells that these variables can derive a string of terminals. After loop in step 2 completes, if $A$ has not been mark then it means $A$ cannot generate any string of terminals hence we reject.
In other case if $A$ is marked, we find out the variables $X$ who can derive a string $Y_1Y_2Y_3...Y_k$ such that at least one of the $Y_i$ is $A$ and rest all can also generate a string of terminals. After all such variables are marked, we check if start symbol $S$ is marked as $\underline{\dot{S}}$. If yes, then it means start symbol can generate a string $w$ whose derivation contains a string in which variable $A$ is present.
The Turing machine $R$ always decides input $\langle G, A\rangle$ as the algorithm terminates.
Thus, the language $U_{TM}$ is decidable. □

## 5 Question 5

Question 5

**Question.** *Consider the problem of determining whether a Turing machine $M$ on an input $w$ ever attempts to move its head left when its head is on the left-most tape cell. Formulate this problem as a language and show that it is undecidable.*

*Solution.* Consider the language $L_{TM}$ defined as

$$L_{TM} = \{\langle M, w\rangle | \\ M \text{ attempts to move its head left when its head is on the left most cell on input } w\} \quad (5)$$

Let's assume that the language $L_{TM}$ is decidable and Turing machine $R$ decides it. $R$ takes $\langle M, w\rangle$ as input and accept if $M$'s tape head attempts to move left when it is on the left most tape cell, else it rejects. Now, consider the following Turing machine $S$ for language $A_{TM}$. It works as:

$S =$ "On input $\langle M, w\rangle$, where $M$ is a Turing machine and $w$ is a string:

1. Construct Turing machine $M_1$ using $M$ and $w$ according to definition given below.
2. Simulate $R$ on input $\langle M_1, w\rangle$
3. If $R$ accepts then accept else reject.

$M_1 =$ "On input $w$:

1. Make left most symbol on tape as "#" and move input string right by one cell on tape.
2. Simulate $M$ on $w$ placing head on second symbol(i.e start of input string) as initial configuration. While simulating $M$ it doesn't allows head to move left of first symbol i.e first cell behaves as start of tape for $M$.
3. If $M$ accepts then try moving head to left of leftmost symbol and accept; else reject.

Here, $M_1$ attempts to move its head left of left most cell on tape iff $M$ accepts $w$. This means $R$ only accepts $M_1$ iff $M$ accepts $w$ else it rejects it. Thus we can say that $S$ decides $A_{TM}$. But this is contradiction to the fact that $A_{TM}$ is undecidable language. Our assumption that $L_{TM}$ is decidable is false.

Hence, $L_{TM}$ is undecidable.

□

## 6 Question 6

Question 6

**Question.** *Consider the problem of determining whether a Turing machine $M$ on an input $w$ ever attempts to move its head left at any point during its computation on $w$. Formulate this problem as a language and show that it is decidable.*

*Solution.* Consider the language $L_{TM}$ defined as

$$L_{TM} = \{\langle M, w\rangle | M \text{ attempts to move left in computation of } w\}$$

Here $L_{TM}$ is a set of encoding of Turing machine $M$ and string w such that head in $M$ tries to move left at any point in computation of $w$.
We will construct as Turing machine $R$ such that on input $\langle M, w\rangle$, it rejects if head always moves right on tape on simulating $M$ on w. We do so by maintaining a second tape on which we maintain a set of transition rules of $M$ applied till yet while simulating $M$ on w on tape 1. Consider the the following cases where $R$ halts and accepts/rejects input.

1. As soon as M transitions using a rule in which head on tape 1 moves lefts, $R$ accepts the input $\langle M, w\rangle$ else reject if $M$ enters into accepts or reject state.
2. $M$'s head continue moving right forever. In this case it will keep reading empty symbols on tape and keep moving right. As the number of states are finite so there will exist a rule in which state repeats and head points to again empty symbol. This is a loop and $M$ will never halt in this case. Using tape 2 we can identify repetition of such rule and thus $R$ rejects such input.

Above case and exhaustive and exclusive, so $R$ will accept the input $\langle M, w\rangle$ when M's head attempts to move left in computation on $w$ else reject. It will always enter into accept or reject state as we have correctly identify the case where $M$ continues moving right and never halts.
Since, $R$ is a Turing machine that decides $L_{TM}$, so we can conclude that $L_{TM}$ is a decidable language. □

## 7 Question 7

Question 7

**Question.** *Let*

$$AMB_{CFG} = \{\langle G\rangle | G \textit{ is an ambiguous CFG } \}$$

*Show that $AMB_{CFG}$ is undecidable via a reduction from PCP.*

*Solution.* To show: $AMB_{CFG}$ is undecidable
Let an instance from PCP be defined as follows

$$\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots, \left[\frac{t_k}{b_k}\right] \right\} \tag{6}$$

Let us generate a corresponding CFG $G$ with the rules:

$$\begin{aligned} &S \to T|B \\ &T \to t_1Ta_1|\ldots|t_kTa_k|t_1a_1|\ldots|t_ka_k \\ &B \to b_1Ba_1|\ldots|b_kBa_k|b_1a_1|\ldots|b_ka_k \end{aligned} \tag{7}$$

where $a_1, \ldots a_k$ are new terminal symbols.
We now need to show the following:

1. **$G$ is ambiguous $\implies$ $P$ has a solution**: This is easy to see since the only two productions of $S$ are via $T$ or $B$. Therefore, the ambiguity can arise only if both these non-terminals yield the same word. Therefore, the following holds true:

$$\begin{aligned} &t_{i_1}t_{i_2}\cdots t_{i_m}a_{i_m}a_{i_{m-1}}\cdots a_{i_1} = b_{i_1}b_{i_2}\cdots b_{i_m}a_{i_m}a_{i_{m-1}}\cdots a_{i_1} \\ \implies &t_{i_1}t_{i_2}\cdots t_{i_m} = b_{i_1}b_{i_2}\cdots b_{i_m} \end{aligned} \tag{8}$$

   This is exactly the same as the solution for a PCP. Therefore, if $G$ is ambiguous, then $P$ has a solution.

2. **$P$ has a solution $\implies$ $G$ is ambiguous**: Consider the solution for $P$. We will now transform the solution into a word in $L(G)$ which has multiple derivations:

$$\begin{aligned} &t_{i_1}t_{i_2}\cdots t_{i_m} = b_{i_1}b_{i_2}\cdots b_{i_m} \\ \implies &t_{i_1}t_{i_2}\cdots t_{i_m}a_{i_m}a_{i_{m-1}}\cdots a_{i_1} = b_{i_1}b_{i_2}\cdots b_{i_m}a_{i_m}a_{i_{m-1}}\cdots a_{i_1} \end{aligned} \tag{9}$$

   As shown above, this string has multiple derivations in $G$. Therefore, the language of $G$ is ambiguous.

Therefore, we have shown that the problem of solving $P$ is same as deciding whether $G$ is ambiguous. Since we have reduced every instance of PCP to a an instance of $AMB_{CFG}$ and we know that PCP is undecidable, therefore, $AMB_{CFG}$ is also undecidable.

*Hence Proved*

*PS: We have taken reference from the hint given in Sipser for this problem.* □

## 8 Question 8

Question 8

**Question.** *In the Silly Post Correspondence Problem (SPCP), the top string in each pair has the same length as the bottom string. Show that SPCP is decidable.*

*Solution.* To Show if a language is decidable, we need to show that for every input, either the input is accepted or reject and the machine halts in finite time.
For the given problem, on analysis we were able to find an algorithm which accepts or rejects an input. Since we can find an algorithm, the language is decidable. The algorithm and the proof of correctness are as follows:

**Algorithm 4** SPCP

```
procedure SPCP($D$)                              ▷ D is set of dominoes
    for $domino$ in $D$ do
        if $domino.num = domino.denom$ then      ▷ numerator and denominator
            return accept
        end if
    end for
    return reject
end procedure
```

**Proof of termination:** We are using a for loop and the number of dominoes is finite so the algorithm terminates.

**Proof of Correctness:** Proof by deduction
For SPCP, $|num| = |denom|$ for all present dominoes. Let us consider a solution $(i_1, i_2, ..., i_m)$ for some set of dominoes. Since numerator = denominator and $|num| = |denom|$ for all dominoes, it can be deduced that $(n_i)_1 = (d_i)_1$, $(n_i)_2 = (d_i)_2$ and so on. This implies that there exists at least one domino such that $n_i = d_i$ for the solution to exist. Since the number of dominoes are finite, both the acceptance and rejection of the input can be decided in finite time. Which means that SPCP is Turing Decidable.

*Hence Proved*

□