# Introduction to Automata and Theory of Computation COL352 - Assignment 3

Satyam Kumar Modi (2019CS50448), Rupanshu Shah (2019CS10395)

March 2022

## Contents

## 1 Infinite Decidable Subset

Let $L$ be an infinite Turing-recognizable language. This implies $L$ is recursively enumerable. Let $M_1$ be Enumerator for language $L$. Let $S = \{w_1, w_2, \cdots\}$ be the sequence of words printed by enumerator $M_1$.
Since $L$ is infinite language, therefore $\forall i \in Z^+,\ \exists w_k$ such that length of $w_k > i$.
Let $S_1 \subseteq S$ defined as $S_1 = \{w_i \in S |\ \forall j < i, j \in Z^+,\ length(w_j) < length(w_i)\}$ i.e., $S_1$ contains words which have more length than all the words before them in the sequence $S$.
Since $S$ is infinite and $\forall i \in Z^+,\ \exists w_k \in S$ such that length of $w_k > i$, therefore $\forall i \in Z^+,\ \exists w_k \in S_1$ such that length of $w_k > i$ i.e., $S_1$ is an infinite subset of $S$.

We claim that $S_1$ is a decidable language.
Let $M_2$ be a TM such that for input $x$, it simulates the enumerator $M_1$, and matches $x$ with the words $w$ enumerator prints.
If $length(x) > length(w)$, it moves to the next word enumerator prints.
Else if $length(x) < length(w)$, it rejects the word $x$.
Else when $length(x) = length(w)$, if $w = x$, it accepts the word $x$; if $w \neq x$, it moves to the next word enumerator prints.
For any input string of length $n$, there are finitely many words in $S$ with length less than or equal to $n$, and the set $S$ is infinite. So, $M_2$ always halts on any input.
Thus $M_2$ decides the language $S_1$.
Therefore, $S_1$ is an infinite decidable subset of arbitrary infinite Turing-recognizable language $L$.
Hence proved.

## 2 Regular

Let us consider a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ such that it cannot write on the portion of the tape containing the input string. We will be using the Myhill-Nerode relation to prove that $L(M)$ is regular.

On an input string $x$, the most general case could be that the tape head keeps going in and out of the input region. We define a map $T : (Q \cup \{\bullet\}) \to Q$. Let the first state to which the machine $M$ goes for the first time just after it gets out of the input region be $q_1$, then we let $T_x(\bullet) = q_1$. For the case where $M$ never enters the non-input region, we assign $T_x(\bullet) = q_{acc}$ if $M$ accepts $x$ and $q_{rej}$ if $M$ don't accept $x$. Now if $M$ is at state $q$ just before entering the input region of the tape from the non-input region and the next time it comes out of the input region, it is in state $q'$, we have $T_x(q) = q'$ . If at some point, $M$ remains inside the input region, then $T_x(q) = q_{acc}$ if $M$ accepts $x$ and $q_{rej}$ if $M$ don't accept $x$.
Now, note that there are only finitely many possibilities for the map $T$ which is $(|Q| + 1)^{|Q|}$. This proves the finite index property in Myhill-Nerode relation.
Now, we show that if $T_x = T_y$ and $x$ is accepted by $M$, then $y$ is also accepted by $M$. The sequence of states in which $M$ goes outside the input region is same for both $y$ and $x$, as $T_x = T_y$. In the final stage, $M$ must have landed up in the final state $q_{acc}$ in the run of $x$ and the same case must have been for $y$(reason being $T_x = T_y$). This proves the Refinement property.
Now, we show the right congruence for $x$ and $y$, let $a \in \Sigma$, we prove if $x \cong y$, then $xa \cong ya$. This is easily noticeable as each time when $M$ crosses the boundary between input region and the non-input region, the states in which $M$ land up is same for both $x$ and $y(x \cong y)$. So, after reading $x$ and $y$ in $xa$ and $ya$, $M$ must be in the same state in both the runs, now since we are reading the same symbol $a$ from the same state in both the runs, we must land up in the same state. When crossing the boundary from non-input region to input region, we must land up in the same state for both $xa$ and $ya$ as the transitions in the non-input region must be same.(because they were in the same state when they first entered the non-input region and the symbols they read are the same which is the blank symbol, so the transitioned state and the new sybol written must be the same). This proves that $T_{xa} = T_{ya}$. So, right congruence property is followed.

So, from Myhill Nerode relation is satisfied for this language $L(M)$. So, $L(M)$ is regular.

## 3 TR iff exists D

*Proof* ($\Leftarrow$)
Let us assume there exists some language $C$, and some decidable language $D$ such that $C = \{x \mid \exists y(\langle x, y\rangle \in D)\}$
Let $M_D$ be a TM for language $D$. Let $M_C$ be a TM, that on input $x$, simulates $M_D$ on $\langle x, y\rangle$ for all strings $y$. If any such $\langle x, y\rangle$ is accepted by $M_D$, then $M_C$ accepts string $x$, else it continues running forever. Thus, if $x \in C$, $M_C$ halts on input $x$ and accepts it. Therefore, $M_C$ is a TM recognizing language $C$. Hence, $C$ is Turing-recognizable.

*Proof* ($\Rightarrow$)
Let $C$ be a Turing-recognizable language. Let $M$ be a TM recognizing language $C$. Define $D = \{\langle x, y\rangle \mid M$ accepts $x$ with accepting configuration history $y\}$.
Language $D$ is decidable as we can have a TM $M_1$ for language $D$ such that for any input $\langle x, y\rangle$, $M_1$ simulates $M$ on $x$ with configuration history $y$. Thus, $M_1$ always halts on any input, and accepts the input $\langle x, y\rangle$ if and only if $M$ accepts $x$ with configuration history $y$. Therefore, $D$ is a decidable language.
Now, $x \in C$ implies $x$ is accepted by $M$ with some configuration history $y$, therefore, $\exists y(\langle x, y\rangle \in D)$.
Similarly, $x \notin C$ implies $x$ is not accepted by $M$ and there is no configuration history corresponding to an accepting run on $x$. Therefore, $\nexists y(\langle x, y\rangle \in D)$.
Thus $D$ is a decidable language satisfying $C = \{x \mid \exists y(\langle x, y\rangle \in D)\}$

Hence proved.

## 4 Usage variable

Let $L_{USAGE} = \{\langle G, A\rangle \mid G$ is a CFG and $A$ is a usable non-terminal in $G\}$. We give the following turing machine R to show decidability of $L_{USAGE}$.

```
R = "On input $\langle G, A\rangle$, where $G$ is a CFG and $A$ is a non-terminal:
    1. Mark all terminal symbols in $G$.
    2. Repeat until no new variables are marked:
    3.     Mark any variable B such that $B \rightarrow U_1U_2..U_k$ such that each $U_1, U_2, .., U_k$ is marked.
    4. If the symbols $S$ or $A$ are unmarked, then reject.
    5. If both symbols $S$ and $A$ are marked:
    6. Collect all rules $B \rightarrow U_1U_2..U_k \in G$ where $B, U_1, .., U_k$ are marked in the set $K$.
    7. For all rules z of the form $B \rightarrow U_1..U_k \in K$ where $U_i = A$ for some $i$:
    8. If check($A$, $K$) is True: Accept
    9. If check($A$, $K$) is False: Reject"
```

**check(X, K)**

```
1. If $X = S$ (starting symbol), return True
2. For all rules z of the form $B \rightarrow U_1..U_k \in K$ where $U_i = X$ for some $i$:
3.     If z is unmarked:
4.         Mark z
5.         If check($B$, $K$) is True, return True
6. return False
```

- **Proof:**
  - We provide an algorithm for this problem. First, we check whether certain symbols are capable of generating terminals or not. This is done in step 1-3 in our algorithm. Next, we check if both the symbols $S$ and $A$ are capable of generating the terminals by checking if they are marked. If not, then we simply reject.

    Else, we now check if there exist some derivation of the form $S \xrightarrow{*} xAy$ where both $x$ and $y$ can derive terminals. For this, we first include those rules in the set $K$ in which all the variables in LHS and RHS of the rule are marked. Now, we start with variable $A$ and look for all the rules which involve $A$ in the RHS (say $B \rightarrow U_1..A..U_k$) and mark that rule so that it do not get repeated(and we are not in a loop), then we go on checking for $B$ in a similiar way, we terminate the recursive checking when $X = S$(starting symbol) in the input to **check(X,K)** function. Since, we have reached $S$ through some production rule of the form $S \rightarrow U_1..U_k$ and since every $U_1, .., U_k$ can produce terminal(as they are in set $K$) and one of these $U_i$ derive $A$, we must have a production of the form $S \xrightarrow{*} xAy$ where both $x$ and $y$ can derive terminals which yields a string $w$ in $L(G)$.

    If we are not able to reach the starting symbol $S$, then we will simply return False and that will lead to **reject** the grammar $G$.

$S \rightarrow U_{01}..B_1..U_{0k}$
$B_1 \rightarrow U_{11}..B_2..U_{1k}$
$B_2 \rightarrow U_{21}..B_3..U_{2k}$
.
.
$B_j \rightarrow U_{j1}..A..U_{jk}$

## 5 Move left decidable

Let language $A$ be defined as $A = \{\langle M, w\rangle |$ TM $M$ attempts to move its head left at some point during its computation on $w\}$
We define a TM $M_A$ that decides language $A$ as follows:
On input $\langle M, w\rangle$, $M_A$ simulates TM $M$ on string $w$ and checks if the head moves to the left at any point. $M_A$ performs the check for only finite number of times ($n = |Q| + |w| + 1$, where $|Q|$ denotes number of states in $M$, $|w|$ denotes length of input)
If the head pointer ever moves left in first $n$ steps, then $M_A$ accepts $\langle M, w\rangle$, else it rejects $\langle M, w\rangle$.
**Claim:** If the head pointer ever moves left, then it moves left atleast once within first $n$ steps; else it never moves left.
*Proof.*
If within the first $|w|$ steps the head ever moves left, then the claim holds.
Now, if the head pointer never moves left within the first $|w|$ steps, then we cross the input after $|w|$ steps and the rest of tape is all blanks.
Now, if within the next $|Q| + 1$ steps, if the head pointer ever moves left, then the claim holds.
If the head pointer only moves right after crossing the input, then the head pointer always points at *blanks* '& '. Since, there can be atmost $|Q|$ distinct pre-images for transitions of the form $\delta : Q \times \{\&\} \to Q \times \Gamma \times \{R\}$, therefore if the head pointer does not move left within $|Q|+1$ steps after crossing the input, then there is a sequence of states $P = q_i, q_{i+1}, q_{i+2}, \cdots, q_j$ within these $|Q| + 1$ steps such that $q_i = q_j$. $M$ repeats this cycle of transitions corresponding to $P$ forever and the head pointer never moves left. Thus, the claim holds.
Thus, if the head pointer ever moves left, then it moves left atleast once within first $n = |w| + |Q| + 1$ steps; else it never moves left. Hence claim proved.

Therefore, for any input $\langle M, w\rangle$, the TM $M_A$ halts within $n = |w| + |Q| + 1$ steps and determines if the head pointer ever moves left on computation of $w$ by TM $M$. Thus, $M_A$ decides language $A$, and hence language $A$ is decidable. Hence proved.

## 6 Move left undecidable

The language $L = \{\langle M, w\rangle |$M attempts to move its head left when its head is on the leftmost tape cell$\}$.

Assume that $L$ is decidable and TM $R$ decides $L$. We use contradiction to prove that $L$ is not decidable.
We construct a TM $S$ that decides the halting problem. Given $\langle M, w\rangle$ as input, we define a new TM $T_M$ that simulates $M$ on $w$ with some changes. $S$ first keeps a special symbol  at the leftmost end. Now, the input $w$ starts from the second cell of the tape. When $M$ tries to move left from the leftmost end of the input, then $T_M$ simulates it as a right move and then a left move, so that it remains in the same cell and the same state. Now, if at some point $M$ accepts $w$, then the head goes to the left until it finds the special symbol and $T_M$ accepts $w$. If $M$ just rejects $w$, then $T_M$ just rejects. Thus, $M$ accepts $w$ iff $T_M$ at some point in the run of input $x$ moves left from the leftmost cell of the input. If $T_M$ do not move left from the leftmost cell of the input means the run was rejected by $M$ or it goes on an infinite loop in which case $R$ rejects.

Formally, we have
S = "On input $\langle M, w\rangle$:
1. Produce the TM $T_M$.
2. Run the TM $R$ that decides $L$ on $\langle T_M, w\rangle$.
3. If $R$ accepts, then accept else reject.

But since the halting problem is not decidable, we arrive at a contradiction that no such TM $R$ exists.

## 7 Ambiguous CFG

We have $AMB_{CFG} = \{\langle G\rangle \mid \text{G is an ambiguous CFG}\}$. To prove that $AMB_{CFG}$ is undecidable, we use the following theorem that if $A$ is mapping reducible to $B$ and $A$ is undecidable then $B$ is also undecidable.

We use $PCP$ as $A$ here. We show how to map an instance of $PCP$ to an instance of $AMB_{CFG}$. Consider a instance of $PCP$ as $P = \{[\frac{a_1}{b_1}], [\frac{a_1}{b_1}], .., [\frac{a_k}{b_k}]\}$. We construct an ambiguous CFG $G$ from the above instance.
$S \to A|B$
$A \to a_1Ac_1|..|a_kAc_k|a_1c_1|..|a_kc_k$
$B \to b_1Bc_1|..|b_kBc_k|b_1c_1|..|b_kc_k$

**Proof:** We prove that there exist a solution for the above $PCP$ problem iff the grammar $G$ is ambiguous.

- **Part-I** Let's say we have a solution for the $PCP$ problem, i.e. there exist a sequence $i_1, i_2, ..i_j$ such that $a_{i_1}a_{i_2}..a_{i_j} = b_{i_1}b_{i_2}..b_{i_j}$,. Now, we can produce the string $a_{i_1}a_{i_2}..a_{i_j}c_{i_1}c_{i_2}..c_{i_j}$ in $G$ by two ways:
  First, we have the rule $S \to A$, then use $A \to a_{i_1}Ac_{i_1}$, $A \to a_{i_2}Ac_{i_2}$, .. ,$A \to a_{i_{j-1}}Ac_{i_{j-1}}$, $A \to a_{i_j}c_{i_j}$ in the same order. This produces the string $a_{i_1}a_{i_2}..a_{i_j}c_{i_1}c_{i_2}..c_{i_j}$.
  Second, we have the rule $S \to B$, then use $B \to b_{i_1}Bc_{i_1}$, $B \to b_{i_2}Bc_{i_2}$, .. ,$B \to b_{i_{j-1}}Bc_{i_{j-1}}$, $B \to b_{i_j}c_{i_j}$ in the same order. This produces the string $b_{i_1}b_{i_2}..b_{i_j}c_{i_1}c_{i_2}..c_{i_j}$.
  Since, $a_{i_1}a_{i_2}..a_{i_j} = b_{i_1}b_{i_2}..b_{i_j}$, we have $a_{i_1}a_{i_2}..a_{i_j}c_{i_1}c_{i_2}..c_{i_j} = b_{i_1}b_{i_2}..b_{i_j}c_{i_1}c_{i_2}..c_{i_j}$. Hence, there are two possible derivation of this string in the grammar $G$ which proves that the grammar is ambiguous.

- **Part-II** Now, suppose that $G$ is ambiguous, then we can say there exist a sequence $i_1, .., i_j$ such that $a_{i_1}..a_{i_j}c_{i_1}..c_{i_j} = b_{i_1}..b_{i_j}c_{i_1}..c_{i_j}$ which implies $a_{i_1}..a_{i_j} = b_{i_1}..b_{i_j}$. Thus, we pick the sequence of dominos as $[\frac{a_{i_1}}{b_{i_1}}], [\frac{a_{i_2}}{b_{i_2}}], .., [\frac{a_{i_k}}{b_{i_k}}]$ and clearly, they are a solution for the PCP problem.

So, we have mapped an instance of the $PCP$ problem to $AMB_{CFG}$. Now, since $PCP$ problem is undecidable implies $AMB_{CFG}$ is undecidable as well.

## 8 Silly PCP

Let $S$ be the set of dominoes of the form $(S_1, S_2)$ where $S_1$ denotes the top string and $S_2$ denotes the bottom string.
Let $M$ be a TM that on input $S$ checks if there is some domino $(S_1, S_2) \in S$ such that $S_1 = S_2$. If there exists such a domino, then it accepts input $S$, else it rejects $S$.
$M$ halts on each input as the input set $S$ is finite, and $M$ just compares the top string $S_1$ and bottom string $S_2$ of each individual domino $(S_1, S_2)$ for equality.
We claim that $M$ decides SPCP.
*Proof.*
If there is a match for set of dominoes $S$, then let $(S_1, S_2)$ be the first domino in the match. Since $(|S_1| = |S_2|)$, therefore $S_1 = S_2$ This implies if there is a match for the set of dominoes $S$, then $\exists (S_1, S_2) \in S$ such that $S_1 = S_2$
Now, if there is a domino $(S_1, S_2) \in S$ such that $S_1 = S_2$, then we have a match consisting of just one domino $(S_1, S_2)$.
Thus $S$ has a match if and only if there is a domino $(S_1, S_2) \in S$ such that $S_1 = S_2$; and $M$ checks whether there is a domino $(S_1, S_2) \in S$ such that $S_1 = S_2$.
Therefore $M$ recognizes SPCP. Since, $M$ halts on every input, therefore $M$ decides SPCP.
Hence, SPCP is decidable.
Hence proved.