COL352
Introduction to Automata and Theory of Computation
Problem Set 4

Nishant Kumar (2019CS50586), Mihir Meena (2019CS10370), Gaurav Jain (2019CS10349)

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

   Solution:
   We know language A is Turing-recognizable iff some enumerator enumerates it.
   Let A be an infinite Turing-recognizable language and E be the enumerator which enumerates it. Consider a subset S of A, $\{w_1, w_2, w_3, ...\}$ enumerated by E such that $|w_i| < |w_j|, \forall i < j$. We need to prove that subset S is infinite and decidable.
   (a) S is infinite.
   We prove this by contradiction. Let us suppose S is finite and w_k is the string with largest length enumerated by E. But we know A is infinite so there must be a string w_l whose length is greater than w_k but not in S. Our assumption contradicts. Hence, S is infinite.
   (b) S is decidable.
   S is decidable if for any input x a Turing machine accepts when x is in S otherwise rejects it. We run the enumerator E and for every output string w, we will compare it with x.
   (i) If w == x then x will be accepted.
   (ii) If $|w| < |x|$, continue running E and compare x with next output string.
   (iii) If $|w| > |x|$, rejects x because all the string having length $\leq |x|$ are already checked.
   Above steps will terminate because x is a finite length input.
   Hence, there exist a infinite decidable subset S of a infinite Turing-recognizable language. $\square$

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

   Solution:
   Let M be a single-tape TM that cannot write on the input portion of the tape. The run of the machine on input string x can be defined as: the tape head stays in the input part and then enter the right side of the input part and again comes back in the input part. This process continues. We define two events- an out event when head moves from input part to non-input part and an in event when head moves from non-input part to input part. Thus, the run can be defined as an alternating sequence of in and out events.

   We define $first-state_x$ as the state M enters after moving to the non-input part for the first time i.e. first out event. In case M doesn't enter the non-input part anytime then $first-state_x = q_{accept}$ if M accepts x and $first-state_x = q_{reject}$ if M rejects x. We also define a function g_x such that for all $q \in Q, g(q,x) = q'$ implies that if M is at state q when it performs an in event then when the machine performs next out event it would be at state q'. If M enters the accept or reject state inside the input part then $g(q,x) = q_{accept}$ or $g(q,x) = q_{reject}$ respectively.

   We define an equivalence relation on the input strings. Two strings x and y are equivalent if $first-state_x = first-state_y$ and for all $q, g(q,x) = g(q,y)$. To show this TM recognize regular language, we want to show this equivalence relation is a Myhill-Nerode Relation. Since the transitions in in and out events are same for x and y, the final states will be same for both x and y. It will also be same for xz and yz and thus, the final states of both strings will be either accepting or rejecting. Finally, the number of choices of $first-state$ and g will be bounded. The equivalence classes are defined using $first-state$ and g and thus they are bounded by $|Q| * |Q|^{|Q|} = |Q|^{|Q|+1}$ which is finite. Hence this relation is a Myhill-Nerode relation and by Myhill-Nerode theorem, the machine T recognize regular languages. $\square$

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

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

Solution:
$\Rightarrow$
If C is a Turing-recognizable language then there must be a TM let's say M that recognizes C.
Let us suppose a language D such that for any $\langle x, y\rangle \in$ D, Turing machine M accepts x in t steps where t $\leq |y|$.

$$D = \{\langle x, y\rangle| \text{ M accepts x in t steps and t } \leq |y|\}$$

For any input $\langle x, y\rangle$,
(i) If x$\in$C and M accepts x in t steps and t $\leq |y|$ then $\langle x, y\rangle \in$ D. If t $> |y|$ then $\langle x, y\rangle \notin$ D
(ii) If x$\notin$ C then either M will not halt or M rejects x. If M does not halt after $|y|$ steps or M rejects x then $\langle x, y\rangle \notin D$.
Hence, from above we can say that D is a decidable language and Turing Machine M decides D.

$\Leftarrow$
To prove C is Turing-recognizable language there should be a TM which accepts string x $\in$C. If D is a Turing decidable language then there must be a TM M which checks whether $\langle x, y\rangle \in D$ or not for all possible string y. If y is found such that $\langle x, y\rangle \in D$ then M accepts else check for another y. So, same TM M can recognize all the string x if $\exists$ y such that $\langle x, y\rangle \in D$. Hence, C is Turing-recognizable language. □

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

   Solution:

   Let $T = \{\langle G, A\rangle \mid G$ is a CFG and A is a usable variable$\}$. A description of T is:
   " On input $\langle G, A\rangle$:

   1. Construct a CFG X which is exactly same as G expect it has A as a terminal and remove all rules with A on left-side. Modify the set of terminals $\Sigma_A = \Sigma \cup A$.
   2. If $L(X)$ is empty then *reject*.
   3. Construct a CFG Y which is exactly same as G expect it has A as its start state i.e. $S = A$.
   4. If $L(Y)$ is empty then *reject*. Otherwise *accept*. "

   If A is usable in G then $S \to^* (\Sigma \cup V)^* A(\Sigma \cup V)^* \to^* \Sigma^* A \Sigma^* \to^* w \in \Sigma^*$ for some w. Since there is a derivation from S that contains A so $L(X)$ is non-empty. Since there is a derivation from A to a substring of w so $L(Y)$ is non-empty. Hence, the machine T will accept.
   If the machine T accepts then both $L(X)$ and $L(Y)$ are non-empty which means that there is a string in $L(Y)$ which is derived from S and contains A. Since $L(X)$ is also non-empty so there exists a string which is derived from A as start state.

5. 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:
   The language for this problem can be given as:

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

   Proof:
   Proof is by contradiction. Assume that L is decidable and let M_{left} be the Turing Machine which decides L. We now construct a Turing Machine H using M_{left} which decides the halting problem. The construction is as follows:
   H = " On input $\langle M, w\rangle$:

   1. Construct a TM H' from H, where H' first moves w over one tape cell to the right and puts a # on the leftmost cell.
   2. Run H' on $\langle M, w\rangle$.
   3. If H' sees # then it moves to the right side and simulates M reaching the leftmost tape cell.
   4. If M halts and accepts on w then H' simulates to move its head left when its head is on the leftmost tape cell.
   5. Run M_{left} on $\langle H', w\rangle$
   6. If M_{left} accepts then *accept*. Otherwise *reject*."

   So, we have a Turing Machine H which is constructed using M_{left} that decides the halting problem. This means that if L is decidable then so is the halting problem. But we know that the halting problem is undecidable. Hence, we have a contradiction. Therefore, L is undecidable.

6. 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:
The language for this problem can be given as:

$$L = \{ \langle M, w \rangle \mid M \text{ ever moves its head left during its computation on w.} \}$$

Now to show that L is decidable, we are required to show that $\exists$ a Turing Machine that halts on all inputs and accepts the words in L and rejects words not in L. Construction of a TM which accepts L is as follows:
M_{left} = " On input $\langle M, w \rangle$:

1. Run the machine for $|Q_M| + |w| + 1$ steps.
2. If M's head moves to the left then *accept*. Otherwise *reject*. "

Claim:
If M's head ever moves left during the computation of w, then it will do so in a computation path of length at most $|Q_M| + |w| + 1$.

Proof:
Let the shortest computation path of M on w ending in a left move be $p = q_0, q_1, ...q_{acc}$. Now, after the state $q_{|w|}$, M would have been reading only blanks as the input string w is finished and it will only move right till q_{acc}. So if there were any cycles between states $q_{|w|}$ and q_{acc}, removing them will result in computation path which still ends on left move. Thus, there is no repeated state in the computation path after $q_{|w|}$ and hence the maximum length of the computation path can be $|Q_M| + |w| + 1$. $\square$

Thus, we have a Turing Machine that takes at most $|Q_M| + |w| + 1$ steps to either accept or reject the given input and hence it accepts L. Therefore, L is decidable.

7. Let

$$AMB_{CFG} = \{\langle G\rangle \mid \text{G is an ambiguous CFG}\}$$

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

Solution:

To show AMB_{CFG} is undecidable we define the reduction $PCP \leq_m AMB_{CFG}$. Let I be an instance of PCP given by $[\frac{w_1}{x_1}]$, $[\frac{w_2}{x_2}]$, ... $[\frac{w_k}{x_k}]$. Consider a CFG G with production rules:

$$S \to A \mid B$$

$$A \to w_1Aa_1 \mid w_2Aa_2 \mid ... \mid w_kAa_k \mid w_1a_1 \mid w_2a_2 \mid ... \mid w_ka_k$$

$$B \to x_1Ba_1 \mid x_2Ba_2 \mid ... \mid x_kBa_k \mid x_1a_1 \mid x_2a_2 \mid ... \mid x_ka_k$$

Here, a_is are new terminal symbols.

Claim:
Instance I of PCP has match iff G is ambiguous.

Proof:
(Instance I has match $\implies$ G is ambiguous)
Since I has a match so there exists a sequence $i_2, i_2, ..., i_n$ such that $w_{i_1}w_{i_2}...w_{i_n} = x_{i_1}x_{i_2}...x_{i_n} = s$. Hence, the string $sa_{i_n}a_{i_{n-1}}...a_{i_1}$ has two derivations via production rules of G.

The two derivations are:

$$S \to A \to w_{i_1}Aa_{i_1} \to w_{i_1}w_{i_2}Aa_{i_2}a_{i_1} \to \ ... \ \to w_{i_1}w_{i_2}...w_{i_n}a_{i_n}...a_{i_2}a_{i_1}$$

$$S \to B \to x_{i_1}Ba_{i_1} \to x_{i_1}x_{i_2}Ba_{i_2}a_{i_1} \to \ ... \ \to x_{i_1}x_{i_2}...x_{i_n}a_{i_n}...a_{i_2}a_{i_1}$$

Thus, G is ambiguous as it contains two derivations for a same string.

(G is ambiguous $\implies$ Instance I has match)
Since G is ambiguous so there exists atleast 2 derivations for some string s. As all production rules contains a_i in the end so all strings of G must end with a_is. The two derivations must be obtained through both A and B.
Let the derivation via A be $w_{i_1}w_{i_2}...w_{i_m}a_{i_m}...a_{i_2}a_{i_1}$ and via B be $x_{i_1}x_{i_2}...x_{i_n}a_{i_n}...a_{i_2}a_{i_1}$.
So, $s = w_{i_1}w_{i_2}...w_{i_m}a_{i_m}...a_{i_2}a_{i_1} = x_{i_1}x_{i_2}...x_{i_n}a_{i_n}...a_{i_2}a_{i_1}$. Since all productions contains equal number of w_i and a_i and equal number of x_i and a_i so the strings $w_{i_1}w_{i_2}...w_{i_m}a_{i_m}...a_{i_2}a_{i_1}$ and $x_{i_1}x_{i_2}...x_{i_n}a_{i_n}...a_{i_2}a_{i_1}$ must contain equal number of a_is as both strings are same and hence, $m = n$.
Therefore, $w_{i_1}w_{i_2}...w_{i_n} = x_{i_1}x_{i_2}...x_{i_n}$ and thus, this sequence is a solution for the instance I of PCP.

Using the above claim, we can reduce an instance of PCP to the problem of checking whether G is ambiguous and thus, $PCP \leq_m AMB_{CFG}$. Since PCP is an undecidable problem so, AMB_{CFG} is also undecidable. □

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

Solution:

Consider an instance I of SPCP $\{[\frac{w_1}{x_1}], [\frac{w_2}{x_2}], ... [\frac{w_k}{x_k}]\}$ where $|w_i| = |x_i|$ for all $i,\ 1 \leq i \leq k$.

Claim:
An instance of SPCP has a match iff there is a domino $[\frac{w_i}{x_i}]$ with $w_i = x_i$.

Proof:
(Instance of SPCP has a match $\implies$ There exists a domino $[\frac{w_i}{x_i}]$ with $w_i = x_i$)
If a instance of SPCP has a match then it must start with some domino d. Since the length of the top and the bottom string is same in all dominos, so, d must also have same length top and bottom strings. Since it is a match so the top and bottom strings match and thus, d have same top and bottom strings.

(There exists a domino $[\frac{w_i}{x_i}]$ with $w_i = x_i \implies$ Instance of SPCP has a match)
If there exists a domino with same top and bottom strings then this single domino can form a trivial solution to the SPCP. Thus, there is a match for an instance of SPCP.

Therefore, the problem of finding match of SPCP reduces to checking whether there is a domino with same top and bottom strings. This problem of checking top and bottom strings of dominos is easily decidable by checking all dominos of an instance of SPCP.
Thus, SPCP is decidable. Hence proved. $\square$