Answers to Practice for the CS456 Final Examination: Part III

1. Let L be any $\mathcal{NP}$-complete language Describe an algorithm for deciding the membership problem for L which has polynomial space complexity.

   We can assume $L \subseteq \Sigma^*$ where Σ is the binary alphabet. Let V_L be a verifier for L such that there is a constant k such that, for any string $w \in \Sigma^*$ of length n, there is a certificate $c \in \Sigma^*$ of langth at most n^k such that V_L accepts the string $w\#c$ in at most n^k steps, and furthermore, that for any $w \notin L$, there is no certificate c for which V_L accepts $w\#c$.

   Let $c_1, c_n, \ldots c_N$ be the enumeration of strings of length at most n^k over Σ in canonical order, where $N = 2^{n^k+1}$. The following program decides L.

```
read $w$
$n = |w|$
$N = 2^{n^k+1}$
$c_1 = \lambda$
if($V_L$ accepts $w\#c_1$ within $n^k$ steps)
   Accept and Halt
For all $i$ from 2 to $N$
   {
     Let $c_i$ be the successor of $c_{i-1}$
     Delete $c_{i-1}$ from memory
     If($V_L$ accepts $w\#c_i$ within $n^k$ steps)
        Accept and Halt
   }
Reject
```

   The memory of the program has size $O(n^k)$ at any given time.

2. Enumeration questions.

   (i) Prove that every recursive language L can be enumerated in canonical order by some machine.

   $L \subseteq \Sigma^*$ for some alphabet Σ. Let $w_1, w_2, \ldots$ be the enumeration of Σ^* in canonical order. The following program enumerates L in canonical order.

```
For i = 1 to $\infty$
   If($w_i \in L$)
     Write $w_i$
```

   Since L is decidable, the condition of the if statement can be evaluated.

   (ii) Prove that every language L that can be enumerated in canonical order by some machine is recursive.

   If L is finite, then L is trivially recursive. Thus, we can assume that L is infinite. Let $w_1, w_2, \ldots$ be an enumeration of L in canonical order. The following program decides L.

```
Read $w$.
For i = 1 to $\infty$
   If($w_i = w$)
     Accept and Halt
```

```
    Else if($w_i > w$) // in canonical order
      Reject and Halt
```

(iii) Prove that every language L accepted by a machine is recursively enumerable.

Let $L \subseteq \Sigma^*$. Let $w_1, w_2, \ldots$ be an enumeration of Σ^* in canonical order. Let M be a machine which accepts L. The following program enumerates L.

```
For t = 1 to $\infty$
   For i = 1 to t
      If($M$ accepts $w_i$ within t steps)
         Write $w_i$
```

If $w \in L$, then $w = w_i$ for some i, and w is accepted by M within j steps for some j. At the t^{th} iteration of the outer loop for $t \geq \max(i, j)$, w will be written. (An enumeration can list the same string any number of times.)

(iv) Prove that every recursively enumerable language is accepted by some machine.

Let $w_1, w_2, \ldots$ be a recursive enumeration of L. The following program accepts L.

```
Read $w$
For i = 1 to $\infty$
   If($w = w_i$)
      Accept and Halt
```

3. DCFL questions.

For both questions, let $L_1 = \left\{a^i b^j c^k \,:\, i \leq j\right\}$, and let $L_2 = \left\{a^i b^j c^k \,:\, j \leq k\right\}$. L_1 and L_2 are both deterministic conext-free languages. Let $\overline{L}_1$ and $\overline{L}_2$ be the complements of L_1 and L_2, respectively, which are also both deterministic context-free languages, since that class is closed under complementation.

(ii) Give two deterministic context-free languages whose intersection is not a DCFL.

$L_1 \cap L_2 = \left\{a^i b^j c^k \,:\, i \leq j \leq k\right\}$ which is not a CFL, hence not a DCFL.

(i) Give two deterministic context-free languages whose union is not a DCFL.

$\overline{L}_1$ and $\overline{L}_2$ are deterministic context free languages. Since $L_1 \cap L_2$ is not a DCFL, its complement, which is $\overline{L}_1 + \overline{L}_2$ by De Morgan's law, is also not a DCFL.

4. $\mathcal{P}$ $\mathcal{NP}$ questions.

(i) Give two $\mathcal{NP}$-complete languages whose intersection is known to be $\mathcal{P}$.

Let L_1 be the set of all members of 4SAT such that the first two terms of each clause are identical, and Let L_2 be the set of all members of 4SAT such that the third and fourth terms of each clause are identical. There is a $\mathcal{P}$ time reduction of 3SAT to L_1 obtained by doubling the first term of each clause, hence L_1 is $\mathcal{NP}$-complete. Similarly, there is a $\mathcal{P}$-time reduction of 3SAT to L_2 obtained by doubling the third term of each clause, hence L_2 is $\mathcal{NP}$-complete. $L_1 \cap L_2$ is all members of 4SAT such that the first term equals the second term and the third term equals the fourth term. We can reduce this language to 2SAT in polynomial time by simply deleting the first and last terms of each clause. Since 2SAT is $\mathcal{P}$, $L_1 \cap L_2$ is $\mathcal{P}$.

There is a much simpler, but less satisfying example. Let L be any $\mathcal{NP}$-complete language over the binary alphabet which does not contain the empty string. There is a simple reduction of the concatenation $0L$ to L: if a string $w \in \Sigma^*$ has a leading zero, just delete that zero; otherwise, map w to the empty string; thus $0L$ is $\mathcal{NP}$-complete. Similarly, $1L$ is $\mathcal{NP}$-complete. $0L \cap 1L = \emptyset$, which is $\mathcal{P}$.

(ii) Give two $\mathcal{NP}$-complete languages whose union is known to be $\mathcal{P}$.

Start with any $\mathcal{NP}$-complete language L over the binary alphabet Σ which contains the empty string λ. Using concatenation, let $L_1 = 0L + 1\Sigma^* + \{\lambda\}$, and let $L_2 = 1L + 0\Sigma^* + \{\lambda\}$. There is a polynomial time reduction R of L to L_1 as follows: for any $w \in \Sigma^*$, let $R(w) = 0w$. Thus L_1 is $\mathcal{NP}$-complete. Similarly, L_2 is $\mathcal{NP}$-complete. $L_1 + L_2 = \Sigma^*$ which is $\mathcal{P}$.

5. Let $\mathcal{N}$ be the natural numbers, that is, the positive integers. Define a function $f : \mathcal{N} \to \mathcal{N}$ which is not recursive.

Recall that $\langle n \rangle$ is the binary numeral for any $n \in \mathcal{N}$. Let L be any undecidable language over the binary alphabet. Define $f : \mathcal{N} \to \mathcal{N}$ as follows.

$$f(n) = \begin{cases} 1 \text{ if } \langle n \rangle = 1w \text{ for } w \in L \\ 0 \text{ otherwise} \end{cases}$$

Since L is undecidable, f cannot be recursive.