Solution of Tutorial 3

Kaushik Mallik

Email: kmallik@mpi-sws.org

Answer 1. (a) **APSPACE** $\subseteq$ **EXP**. This direction is trivial. Any alternating TM which uses polynomial space can be simulated by a DTM running in exponential time.

(b) **EXP** $\subseteq$ **APSPACE**. Let $L \in$ **EXP**. Then there is a DTM M which runs in time $O(2^{n^c})$, for some constant c, to decide L. Consider the configuration graph $G_{M,x}$ of M on input x. The machine accepts x if there is a path from the initial configuration to the accepting configuration of length 2^{n^c}. Such a path exists if and only if there *exists* configurations $C_1, \dots, C_{2^{n^c-1}}$ s.t. *for all* $i \in [2^{n^c-1}]$, C_{i+1} takes at most 2^{n^c-1} steps from C_i. This quantification alternation can be can be realized by an alternating TM D. Since space can be reused, D just needs to keep track of the last configuration visited and to keep a counter. Similar to Cook-Levin theorem, each configuration can be represented by encoding the contents local to the tape head. Hence, D can be simulated to use only polynomial space.

Answer 2. First, it will be shown that any arbitrary language in Σ_i^p can be reduced in polynomial time to Σ_iSAT. Let $L \in \Sigma_i^p$ be any arbitrary language. Then by definition, there is a polynomial time TM M and a polynomial q s.t.

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)}.\forall u_2 \in \{0,1\}^{q(|x|)} \dots Q_i u_i \in \{0,1\}^{q(|x|)}.M(x, u_1, u_2, \dots, u_i) = 1.$$

Following similar construction as in the proof of Cook-Levin theorem, one can use the configuration graph of the TM M, subjected to the input $\langle x, u_1, u_2, \dots, u_i \rangle$, to create a formula φ s.t. M accepts the input if and only if φ is satisfiable (i.e. $\varphi \in$ SAT). It is known that this reduction can be done in polynomial time.

Furthermore, from the definion of Σ_iSAT, it is easy to see that Σ_iSAT is itself in Σ_i^p. Hence Σ_iSAT is Σ_i^p-complete.

Answer 3. (a) First, it will be shown that any language in **DP** is reduced to $SAT - UNSAT$ in polynomial time. Let $L \in$ **DP**. Then by definition, there exist two languages $L_1 \in$ **NP** and $L_2 \in$ **coNP** s.t. the following holds:

$$x \in L \Leftrightarrow x \in L_1 \wedge x \in L_2. \tag{1}$$

Using Cook-Levin theorem, one can reduce L_1 and L_2, subjected to input x, to SAT and UNSAT (by reducing $\overline{L_2}$ to SAT instance) instances in polynomial time; let ϕ_1 and ϕ_2 represent the corresponding boolean formulae respectively. Then we can say that

$$x \in L \Leftrightarrow \phi_1 \in \text{SAT} \wedge \phi_2 \in \text{UNSAT},$$

which by definition is an instance of the language SAT − UNSAT.

Moreover, SAT − UNSAT is itself in **DP** by definition. Hence SAT − UNSAT is **DP**-complete under ploynomial time reduction.

(b) To prove that $EXACT - INDSET$ is **DP**-complete, we use the property that $INDSET$ is

NP-complete. Following is the definition of $INDSET$ and $\overline{INDSET}$:

$$INDSET = \{\langle G, k\rangle \mid \text{Graph } G \text{ has an independent set of size } \geq k\}$$
$$\overline{INDSET} = \{\langle G, k\rangle \mid \text{Graph } G \text{ does not have an independent set of size } \geq k\}$$

Then the language $EXACT - INDSET$ can be defined as:

$$EXACT - INDSET = \{\langle G, k\rangle \mid \langle G, k\rangle \in INDSET \wedge \langle G, k+1\rangle \in \overline{INDSET}\}.$$

Now any language which is in **DP** has two associated languages in **NP** and **coNP** respectively (as in Eqn. (1)). Since $INDSET$ is **NP**-complete and consequently $\overline{INDSET}$ is **coNP**-complete, hence L_1 and L_2 reduces in polynomial time to $INDSET$ and $\overline{INDSET}$ respectively. Hence $EXACT - INDSET$ is **DP**-hard.

Moreover, since $INDSET$ and $\overline{INDSET}$ are themselves in **NP** and **coNP**, hence $EXACT - INDSET$ is in **DP** itself. Hence completeness is proven.

Answer 4. (a) First it will be shown that using Shannon's decomposition, every n-ary function can be computed using a circuit of size $O(2^n)$. Fig. 1 shows the circuit in the form of a tree. Note that the $\wedge$ and $\vee$ gates are located at interleaving depths. We can simply count the number of gates by observing the tree:

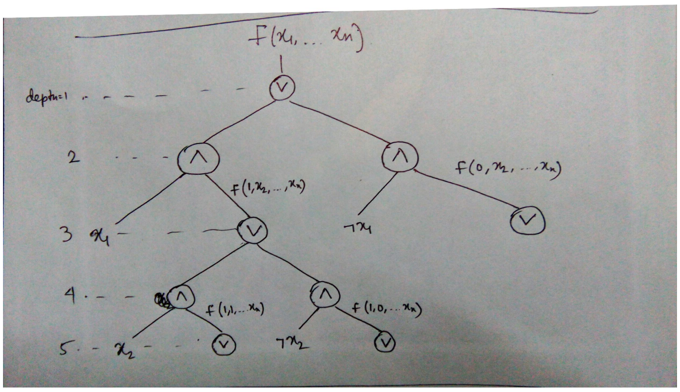

Figure 1: Circuit realizing a function f using Shannon's decomposition

The total number of $\wedge$ gates in the circuit =

$$2^1 + 2^2 + \ldots + 2^n = \frac{2 \times 2^{n+1} - 2}{2 - 1} = 2^{n+2} - 2.$$

Total number of $\vee$ gates =

$$2^0 + 2^1 + \ldots + 2^n = \frac{1 \times 2^{n+1} - 1}{2 - 1} = 2^{n+1} - 1.$$

Total number of $\neg$ gates = n. Hence, size of the circuit = $2^{n+2} + 2^{n+1} + n - 3 = O(2^n)$.

(b) Now the second part will be proved, i.e. every n-ary function can be realized by a circuit of size $O(2^n/n)$.

1. **Claim 1.** For each l, there is a circuit of size $O(2^{2^l})$ with 2^{2^l} outputs which computes all l-ary Boolean functions simultaneously.

Proof. The claim will be proved by induction.

Base case: $i = 1$. There are four possible functions of the form $\{0,1\} \to \{0,1\}$, as in the following:

$$\begin{array}{ll} f_1(x_1) = 0 & f_2(x_1) = 1 \\ f_3(x_1) = x_1 & f_4(x_1) = \neg x_1 \end{array}$$

These functions can be simultaneously realized by a circuit with four outputs and just one $\neg$ gate. Hence the Claim holds for $i = 1$. Inductive step: let, the Claim holds for $i = l$. It will be shown that the Claim will hold for $i = l + 1$ as well.

Given an $(l+1)$-ary function $f(x_1, \ldots, x_{l+1})$, we can use Shannon's decomposition:

$$\begin{aligned} f(x_1, \ldots, x_{l+1}) &= x_1 \wedge f(1, x_2, \ldots, x_{l+1}) \vee \neg x_1 \wedge f(0, x_2, \ldots, x_{l+1}) \\ &= x_1 \wedge g(x_2, \ldots, x_{l+1}) \vee \neg x_1 \wedge g(x_2, \ldots, x_{l+1}). \end{aligned}$$

g being an l-ary function, by induction hypothesis, there exists a $O(2^{2^l})$ size circuit C which outputs simultaneously the 2^{2^l} possible forms of g. Using C as a building block, we can realize f by a circuit of size $\left(O\left(2^{2^l}\right)\right)^2 + 1 = O\left(2^{2^{l+1}}\right) + 1 = O\left(2^{2^{l+1}}\right)$. □

2. Next, it will be shown that for a choice of k s.t. $2^k > 2^{2^{n-k}}$, f can be computed using a circuit of size $s = O(2^n/n)$. Set $l = log_2(k)$. Then by Claim 1, there is a circuit of size $O(2^{2^l}) = O(2^{2^{log_2(k)}}) = O(2^k)$ which computes all l-ary functions and produces 2^k outputs simultaneously. This circuit may be used to compute the functions $h_1, \ldots, h_{2^k}$ simulataneously, provided the number of inputs l is greater than $n - k$ to accomodate all the $(n-k)$ inputs of $h_1, \ldots, h_{2^k}$. But this condition holds always because of the choice of k: $2^k > 2^{2^{n-k}} \Rightarrow k > 2^{n-k} \Rightarrow l > n - k$.

   We will find an optimum k for attaining minimum circuit complexity. Consider the following derivation:

$$\begin{aligned} & l > n - k \\ \Rightarrow\ & k > n - l \\ \Rightarrow\ & k > n - log_2(k) \\ & \quad > n - log_2(n) \end{aligned} \tag{2}$$

   where the last inequality comes from $k < n \Rightarrow log_2(k) < log_2(n)$.

   Now let us compute the total size of the resulting circuit. This is given by

$$s + O(2^k) = O(2^k) + O(2^k) = O(2^k).$$

   Then by Inq (2), the minimum complexity for optimum choice of k is given by

$$O\left(2^{n - log_2(n)}\right) = O\left(\frac{2^n}{2^{log_2(n)}}\right) = O\left(\frac{2^n}{n}\right).$$

Answer 5. $\mathbf{P} = \mathbf{NP}$ implies $PH = \mathbf{P}$. Since $\mathbf{P} \subset \mathbf{EXP}$, hence using $\mathbf{P}$ as oracle does not add any extra power to the class $\mathbf{EXP}$; the querries to the oracle $\mathbf{P}$ can be simulated by the TM (running in exponential time) itself with no extra overhead. Hence, $\mathbf{EXP} = \mathbf{EXP}^{\mathbf{P}} = \mathbf{EXP}^{PH}$.

–Incomplete–

Answer 6. **Claim:** Iterated addition is in $\mathbf{NC}^1$.

Proof. Given k n-bit numbers $a_1, \ldots, a_k$, one can successiely use the constant depth circuit for addition to create a logarithmic depth $(O(log(k \times n)))$ circuit as follows:

$$\sum_{i=1}^{k} a_i = ((a_1 + a_2) + (a_3 + a_4)) \ldots ((a_{k-3} + a_{k-2}) + (a_{k-1} + a_k))$$

Hence iterated addition is in $\mathbf{NC}^1$. □

Claim: multiplication of two n-bit numbers is in $\mathbf{NC}^1$.

Proof. Given two n-bit numbers a, b, multiplication can be implemented with the help of iterated addition as in the following:

$$a \times b = \sum_{i=1}^{n} a \times b_i \times 2^i$$

Each term $(a \times b_i \times 2^i)$ can be realized using constant depth circuit, and the sum can be implemented using logarithmic depth circuit. □

Answer 7. Consider the following undecidable language:

$L := \{1^k \mid \text{binary representation of } k \text{ represents an encoding } < M, x > \text{ s.t. the TM M halts on input } x\}$.

Think of a real number $\rho \in [0, 1]$ as an advice string for a PTM N, s.t. the k-th bit of the binary representation of ρ is 1 iff $1^k \in L$. However the exact value of ρ is unknown to the PTM N, which only has access to a biased coin coming up with head with probability ρ. The question is then given input 1^k, how to recover the k-th bit of ρ by repeatedly tossing the coin.

The question appeared in previous year's homework problem set, and the following solution was presented by Oliver Bachtler:

$L = \{1^k \mid M_k \text{ halts on input } 1^k\}$ is undecidable. Let

$$\rho := \sum_{k=1}^{\infty} 2^{-2k} + \sum_{k=1}^{\infty} b_k 2^{-2k+1} \in [0, 1].$$

Note that the last part is true, because the right hand side is less than or equal to $\sum_{i=0}^{\infty} 2^{-i} - 1 = 2 - 1 = 1$ and as such it is bounded from above and is monotone increasing, hence it converges.

Let M be a PTM that can flip a coin with a chance of ρ to land on heads. We claim that M can decide L, but before we describe how M works we make one observation.
Let X_n be the random variable that counts the number of times heads comes up in n flips of the ρ-biased coin. Then $E[X_n] = n\rho$ and $Var[X_n] = n\rho(1-\rho)$. By the Chebyshev's inequality (or however you want to spell it) we get

$$P\left[\left|\frac{X_n}{n} - \rho\right| \geq \varepsilon\right] = P\left[|X_n - n\rho| \geq n\varepsilon\right] \leq \frac{n\rho(1-\rho)}{n^2\varepsilon^2} = \frac{\rho(1-\rho)}{n\varepsilon^2}.$$

This gives us that

$$P\left[\left|\frac{X_n}{n} - \rho\right| < \varepsilon\right] \geq 1 - \frac{\rho(1-\rho)}{n\varepsilon^2}. \tag{1}$$

We make use of this as follows: On input 1^k M flips $n = \frac{\rho(1-\rho)}{\varepsilon^2} \cdot 3$ coins, where $\varepsilon = 2^{-2k}$. If, as before, X_n is the amount of times heads is flipped then M returns the $(2k-1)$-st bit of $\frac{X_n}{n}$ (to be precise, the $(2k-1)-st$ bit after the comma). Plugging our values for n and ε into (1), we get

$$P\left[\left|\frac{X_n}{n} - \rho\right| < \varepsilon\right] \geq 1 - \frac{1}{3} = \frac{2}{3}. \tag{2}$$

Now let us make sure this does the trick.
Case 1 $L(x) = 1$: Let X be a number such that the $(2k-1)-st$ bit is 0. Let x' correspond to ρ on the first $2k-2$ bits followed by only ones and ρ' corresponds to ρ on the first $2k$ bits, followed by only zeroes. This gives us that $x \leq x' \leq \rho' \leq \rho$ and consequently we get

$$|x-\rho| \geq |x'-\rho'| = 2^{-2k+1} - \sum_{i=1}^{\infty} 2^{-(2k+i)} = 2^{-2k}. \tag{3}$$

Using $x = \frac{X_n}{n}$ in (3) together with the fact that the $(2k-1)$-st bit of ρ is one we get

$$\left|\frac{X_n}{n} - \rho\right| < 2^{-2k} \Rightarrow (2k-1)\text{-st bit of } \frac{X_n}{n} \text{ is one.} \tag{4}$$

After gathering together all the necessary parts, it is time to check the actually required property:

$$\begin{aligned} Pr[M(x) = 1] &= Pr\left[(2k-1)\text{-st bit of } \frac{X_n}{n} \text{ is } 1\right] \\ &\overset{(4)}{\geq} Pr\left[\left|\frac{X_n}{n} - \rho\right| < 2^{-2k}\right] \\ &\overset{(2)}{\geq} \frac{2}{3}. \end{aligned}$$

Case 2 $L(x) = 0$: This case works analogously. In the same way as above we get

$$\left|\frac{X_n}{n} - \rho\right| < 2^{-2k} \Rightarrow (2k-1)\text{-st bit of } \frac{X_n}{n} \text{ is zero} \tag{5}$$

and again

$$\begin{aligned} Pr[M(x) = 0] &= Pr\left[(2k-1)\text{-st bit of } \frac{X_n}{n} \text{ is } 0\right] \\ &\overset{(5)}{\geq} Pr\left[\left|\frac{X_n}{n} - \rho\right| < 2^{-2k}\right] \\ &\overset{(2)}{\geq} \frac{2}{3}. \end{aligned}$$

This covers both cases and shows that M in fact computes L.